

D7.2 – Integration Report v2

Deliverable No.	D.7.2	Due Date	30- JUN-2021 (version sent 6- JUL-2021, incurring in a 6-day delay).
Type	Report	Dissemination Level	Public
Version	2.0	Status	Final
Description	This document will report about the integration activities in two versions. It will include test plans (individual, module and integration), the different integration environments (lab, real scenario, real data). It will report of all the pilot integration activities including technical, organizational and operational aspects. Associated task: T7.1, T7.2, T7.3, T7.4 and T7.5.		
Work Package	WP7		

Authors

Name	Partner	e-mail
Ignacio Lacalle	1- UPV	iglaub@upv.es
Benjamin Molina	1- UPV	benmomo@upvnet.upv.es
Ismael Torres	2- PRO	itorres@prodevelop.es
Miguel Ángel Llorente	2- PRO	mllorente@prodevelop.es
Alvaro Garcia Faura	3- XLAB	alvaro.garcia.faura@xlab.si
Gilda De Marco	4- INSIEL	gilda.demarco@insiel.it
Paolo Casoto	4- INSIEL	paolo.casoto@gmail.com
Charles Garnier	5- CATIE	c.garnier@catie.fr
Erwan Simon	5- CATIE	e.simon@catie.fr
Marc Despland	6-ORANGE	marc.despland@orange.com
Teodora Milosevic	8-MEDRI	teodora.milosevic@medri.uniri.hr
Stjepan Piličić	8 MEDRI	stjepan.pilicic@gmail.com
Tamara Corsano	9- SDAG	t.corsano@sdag.it
Eleonora Anut	9- SDAG	e.anut@sdag.it
Cinzia Ninzatti	9- SDAG	c.ninzatti@sdag.it
Eirini Tserga	10- ThPA S.A.	etserga@thpa.gr
Grigoris Dimitriadis	10- ThPA S.A.	gdimitriadis@thpa.gr
Stefano Bevilacqua	12-APT	stefano.bevilacqua@porto.trieste.it
Fabrice Klein	13-GPMB	F-Klein@bordeaux-port.fr
George Litainas	14-PEOPLE	glitainas@people-t.com
Leonidas Pitsikas	14-PEOPLE	lpitsikas@peoplethinkbeyond.com

History

Date	Version	Change
19-MAY-2020	0.1	Draft of the ToC proposed
1-MAY-2021	0.2	Agreement on a final ToC
10-MAY-2021	0.3	Sections assignment confirmation and first contributions
26-JUN-2021	0.4	Most of the content ready
30-JUN-2021	0.5	Rearrangement of content and request of modification from Internal Reviewers
6-JUL-2021	1.0	Submission to EC after a review of the document by the TC and PC

Key Data

Keywords	Use Case, Ports, Test Plans, Validation
Lead Editor	Ismael Torres - P02 PRO
Internal Reviewers	Eirini Tserga (P10 THPA), Leonidas Pitsikas (P14 PEOPLE)

Abstract

This deliverable has been created in the context of the Work Package 7 (Pilot trials integration, deployment and evaluation) of the H2020-funded project PIXEL (Grant No. 769355).

The present document is the second version of the “Integration Report” (D7.1) and it mainly aims at explaining the activities carried out to integrate the different components of the PIXEL platform developed in WP4, WP5 and WP6, the development of the four use cases (including the deployment of all modules in ports’ infrastructures) and the application of the Port environmental Index (PEI) in all of them. Moreover, it explains the results, processes and findings after successfully achieving those deployments.

The work explained in the document is structured following the use-cases that were defined back in WP3 (deliverables D3.3 and D3.4), making use of the models or predictive algorithms developed during WP4 (WP5 in case of the PEI).

Each of the pilots has been associated to one of the ports of the project and has tried to solve the real need of the port. The energy management trial will be implemented in the Port of Bordeaux. The objective of the pilot has been to understand the energy needs and the impact of the different port activities in terms of energy consumption. An energy model has been used to calculate the prediction of the demand energy and energy consumption based on the Planned Port Activity. Thanks to these models, the port is able to determine its energy needs. The Intermodal transport trial will be implemented in the Port of Monfalcone. The objective of the pilot is to improve intermodal transport, pushing connection by rail between Port of Monfalcone and SDAG to increase efficiency of logistics. Two models related to truck congestion at different areas/gates are used to predict traffic congestion. This information is being used by the port to organize its activity in order to reduce the parking and road congestion. The Port-City integration trial will be implemented in the Port of Piraeus and Thessaloniki. For them, it is important to improve coexistence with the city and to minimise the environmental and traffic impact due to port activity. In these pilots models related with air quality and noise have been used to calculate how the port activity affects these environmental factors. On the other hand, the traffic congestion caused by Ports is also analysed by a traffic prediction model. These models provide useful data to make decisions in order to become greener ports and reduce their impact in cities. The Port Environmental Index is one of the main results of the project and it has been applied successfully in the four PIXEL Ports. Thanks to the PEI the ports have a useful tool to measure their environmental impact along the time to reduce their environmental impact, allowing them to compare their impact with similar ports. The main conclusion after executing the PEI in the PIXEL ports is that the ships are the most determining polluting factor, and therefore where ports have to put more emphasis in order to reduce their environmental impact.

Statement of originality

This document contains material, which is the copyright of certain PIXEL consortium parties, and may not be reproduced or copied without permission. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

The information contained in this document is the proprietary confidential information of the PIXEL consortium (including the Commission Services) and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the project consortium as a whole nor a certain party of the consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk and accepts no liability for loss or damage suffered by any person using this information.

The information in this document is subject to change without notice.

The content of this report reflects only the authors’ view. The Innovation and Networks Executive Agency (INEA) is not responsible for any use that may be made of the information it contains.

Table of contents

Table of contents	5
List of tables	8
List of figures	9
List of acronyms	10
1. About this document	12
1.1. Deliverable context	12
1.2. Rationale behind the structure	13
2. PIXEL Validation Methodology	15
2.1. Global assumptions and flow	15
2.2. PIXEL platform installation block	16
2.3. Data integration design block	18
2.4. NGSi agents validation block	18
2.5. Models/predictive algorithms validation block	20
3. Testing PIXEL deployments	22
3.1. End-to-end scenario testing	22
3.2. Formal Testing	22
4. Energy management trial – Port of Bordeaux	23
4.1. Context review	23
4.2. Platform infrastructure	24
4.3. Data sources	24
4.3.1. Vessel calls (planned)	24
4.3.2. Vessel calls (historic)	24
4.3.3. Tide level	25
4.3.4. Air pollution ATMO	26
4.3.5. Weather station	26
4.3.6. Noise/PM10/Light	27
4.3.7. MARPOL Annexes – Data of ships	27
4.3.8. Waste terminal – Terminal of Bassens	28
4.3.9. Estimation of photovoltaic production per day	29
4.3.10. Port parameters	29
4.4. Functionalities provided, models and algorithms	30
4.4.1. PAS model	31
4.4.2. ETD Predictive Algorithm	35
5. Intermodal Transport trial – Port of Monfalcone	35
5.1. Context review	35
5.2. Platform infrastructure	36
5.3. Data sources	37

5.3.1.	SILI	37
5.3.2.	Available spots in the parking of SDAG	37
5.3.3.	Smart Camera	38
5.3.4.	Vessel Call.....	39
5.4.	Functionalities provided, models and algorithms	39
5.4.1.	PAS model.....	40
5.4.2.	Intermodal transport model	41
5.4.3.	Traffic Predictive Algorithm	43
6.	Port-city integration trial – Port of Thessaloniki	47
6.1.	Context review.....	47
6.2.	Platform infrastructure	47
6.3.	Data sources	47
6.3.1.	Vessel Calls	47
6.3.2.	Gates Traffic Data.....	48
6.3.3.	Wind Data.....	49
6.4.	Functionalities provided, models and algorithms	50
6.4.1.	Traffic Predictive Algorithm	51
6.4.2.	Noise Model	54
6.4.3.	Air pollution model.....	55
7.	Port-city integration trial – Port of Piraeus.....	56
7.1.	Context review.....	56
7.2.	Platform infrastructure	57
7.3.	Data sources	57
7.3.1.	Vessel Calls	57
7.3.2.	Dark Sky.....	57
7.3.3.	Cruise Ships.....	58
7.3.4.	Here maps.....	59
7.4.	Functionalities provided, models and algorithms	60
7.4.1.	Traffic Predictive Algorithm	61
7.4.2.	Noise Model	62
7.4.3.	AERMOD.....	64
8.	Transversal trial: Port Environmental Index.....	66
8.1.	PEI development and integration methodology.....	66
8.2.	Story and operative details	67
8.3.	Definition of the model (configuration).....	68
8.4.	PEI deployment in PIXEL: Data used and NGSI agents developed.....	74
8.4.1.	GPMB: Port of Bordeaux	74
8.4.2.	APT: Port of Monfalcone	76
8.4.3.	THPA: Port of Thessaloniki	78

8.4.4. PPA: Port of Piraeus	80
8.5. Visualisation of results.....	81
8.6. Conclusions after technical deployment of PEI.....	83
9. Conclusions	85
Appendix A– PIXEL Data Models.....	86
Appendix B – HOPU Sensor Acquisition effort process	89
Appendix C – NGSI Agents Cookbook.....	93
Appendix D – Ping scenario outcomes.....	101
Appendix E – AERMOD worst-case scenario (THPA).....	103
Appendix F –PEI Report example (PDF)	117

List of tables

Table 1. Deliverable context	12
Table 2. Data source “Vessel calls planned” Port of Bordeaux	24
Table 3. Data source “Vessel calls history” Port of Bordeaux	24
Table 4. Data source “Tide Level” Port of Bordeaux	25
Table 5. Data source “Air Pollution ATMO” Port of Bordeaux	26
Table 6. Data source “Weather Station” Port of Bordeaux	26
Table 7. Data source “Noise/PM10/Light” Port of Bordeaux	27
Table 8. Data source “MARPOL Data Ships” Port of Bordeaux	27
Table 9. Data source “Waste terminal Bassens” Port of Bordeaux	28
Table 10. Data source “Estimation of PV production per day in the port” Port of Bordeaux	29
Table 11. Data source “Port parameters” Port of Bordeaux	29
Table 12. Functionalities deployed in GPMB – task T7.2	30
Table 13. Model “PAS Model” Port of Bordeaux	31
Table 14. Sample of PVGIS model output used in the renewable balance functionality	34
Table 15. Predictive Algorithm “ETD” Port of Bordeaux	35
Table 16. Data source “SILI” Port of Monfalcone	37
Table 17. Data source “SDAG_ACS” Port of Monfalcone	37
Table 18. Data source “Smart Camera” Port of Monfalcone	38
Table 19. Data source “Vessel Call” Port of Monfalcone	39
Table 20. Functionalities deployed in APT – task T7.3	40
Table 21. Model “PAS Model” Port of Monfalcone	40
Table 22. Model “Intermodal transport model” Port of Monfalcone	41
Table 23. Algorithm “Prediction algorithm” Port of Monfalcone	44
Table 24. Data source “Vessel calls planned” Port of Thessaloniki	47
Table 25. Data source “Gates Traffic data” Port of Thessaloniki	48
Table 26. Data source “Wind data” Port of Thessaloniki	49
Table 27. Functionalities deployed in THPA – task T7.4	50
Table 28. Algorithm “Prediction algorithm” Port of Thessaloniki	52
Table 29. Model “Noise model (import facility)” Port of Thessaloniki	54
Table 30. Model “Air pollution model (import facility)” Port of Thessaloniki	55
Table 31. Data source “Vessel calls planned” Port of Piraeus	57
Table 32. Data source “Cruise ships” Port of Piraeus	58
Table 33. Data source “Here maps” Port of Piraeus	59
Table 34. Data source “Meteo” Port of Piraeus	59
Table 35. Functionalities deployed in PPA – task T7.4	60
Table 36. “Traffic” Predictive Algorithm Port of Piraeus	61
Table 37. Model “Air Pollution model” Port of Monfalcone	64
Table 38. Analysis of the frequency of the data and the nature of PEI NGSI agents in GPMB	74
Table 39. Analysis of the frequency of the data and the nature of PEI NGSI agents in GPMB	75
Table 40. Analysis of the frequency of the data and the nature of PEI NGSI agents in APT	76
Table 41. Analysis of the frequency of the data and the nature of PEI NGSI agents in APT	77
Table 42. Analysis of the frequency of the data and the nature of PEI NGSI agents in THPA	78
Table 43. Analysis of the frequency of the data and the nature of PEI NGSI agents in PPA	80
Table 44. Analysis of the frequency of the data and the nature of PEI NGSI agents in APT	80
Table 45. FIWARE Data Models used on PIXEL list	87
Table 46. PIXEL specific Data Model list	87
Table 47. Deployment of the end-to-end scenario steps	101
Table 48. Ships Air emissions rate	104
Table 49. Terminal Air emissions rate	104
Table 50. Total Air emissions rate	104
Table 51. AERMAP data for THPA	105
Table 52. AERMED data for THPA	109

Table 53. AERMOD data for THPA	111
Table 54. AMBIENT AIR QUALITY STANDARDS FOR THE EUROPEAN UNION	116

List of figures

Figure 1. Global assumptions in PIXEL model's validation methodology.....	15
Figure 2. Global flow for the use-case deployment and validation methodology using PIXEL.....	16
Figure 3. PIXEL platform installation block flow of states	16
Figure 4. PIXEL NGSI agents breakdown.....	19
Figure 5. PIXEL NGSI agents validation block flow of states.....	19
Figure 6. Port parameters being introduced before running the PAS (sample).....	32
Figure 7. Gantt visualisation of the results of the PAS model	33
Figure 8. Energy consumption evolution out of PAS model (cumulative)	33
Figure 9. Energy consumption evolution out of PAS model (instant).....	34
Figure 10. Renewable balance functionality visualisation.....	34
Figure 11. ETD Predictive Algorithm results visualisation in PIXEL.....	35
Figure 12. APT Intermodal transport model - parking lot expected occupancy output.....	42
Figure 13. APT Intermodal transport model - effective flow output.....	42
Figure 14. APT Intermodal transport model max flow output	43
Figure 15. Port gates in ASPM. Vehicle tracking	44
Figure 16. APT Traffic prediction algorithm - port gates and city traffic volume map.....	45
Figure 17. APT Traffic prediction algorithm - port gates 7 days traffic monitor.....	46
Figure 18. APT Traffic prediction algorithm - traffic information with predictions.....	46
Figure 19. Vessels calls THPA data source	48
Figure 20. Traffic at the gates data THPA	49
Figure 21. Wind data source THPA	49
Figure 22. Port gates in THPA. Vehicle tracking.....	51
Figure 23. THPA traffic congestion illustration.....	51
Figure 24. THPA prediction algorithm. Actions taken	52
Figure 25. THPA traffic prediction. Results visualizations in the Dashboard	52
Figure 26. THPA Prediction algorithm - port gates traffic prediction.....	53
Figure 27. THPA Prediction algorithm - congestion status map.....	53
Figure 28. THPA Prediction algorithm - current traffic	54
Figure 29. THPA Noise model - Noise level map layer	55
Figure 30. THPA Air pollution model - Air pollution level map layer	56
Figure 31: PPA Velocity prediction	61
Figure 32. ATP Noise KML importation form	63
Figure 33. ATP Noise model - KML representation	64
Figure 34. APT Air pollution model - PNG output of AEROMOD execution.....	65
Figure 35. PEI overview	66
Figure 36. PEI Tree configuration from the Dashboard	69
Figure 37. PEI Tree Configuration (default values)	70
Figure 38. PEI Forms. Waste Terminal	71
Figure 39. PEI Forms. Wastewater Terminal.....	71
Figure 40. PEI forms. Waste Port Authority	72
Figure 41. PEI Forms. Wastewater Port Authority.....	72
Figure 42. PEI Reliability Ratings (Dashboard).....	74
Figure 43. List of PEI executions in THPA (year 2020).....	81
Figure 44. PEI results for THE- year 2020 (Part 1).....	82
Figure 45. PEI results for the year 2020 (Part 2).....	83
Figure 46. HOPU cloud approach	89
Figure 47. HOPU local approach	89

Figure 48. HOPU cloud approach adaptations for THPA.....	90
Figure 49. Needed noise values for THPA	91
Figure 50. Needed light (illuminance) values for THPA.....	91
Figure 51. Direct integration through MQTT	92
Figure 52. Emission areas	103
Figure 53. PEI Report example	119

List of acronyms

Acronym	Explanation
AERMOD	AIR Quality Dispersion Model
AIS	Automatic Identification System
API	Application Programming Interface
ASPM	CCIAA DI GORIZIA - AZIENDA SPECIALE PER IL PORTO DI MONFALCONE
BTEX	Benzene, Toluene, Ethylbenzene and Xylene
CO	Carbon Monoxide
CSV	comma-separated values
DAL	Data Acquisition Layer
EE	Electric energy
eKPI	Environmental Key Performance Indicator
ETD	Estimated Time of Departure
FTP	File Transfer Protocol
GPMB	Grand Port Maritime de Bordeaux - Port of Bordeaux
IH	Information Hub
INEA	Innovation and Networks Executive Agency
IoT	Internet of Things
IoTRL	IoT-Readiness level
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
KPI	Key Performance Indicators
LED	Light Emitting Diode
LNG	Liquified Natural GAS
MQTT	Message Queuing Telemetry Transport
NGSI	Next Generation Service. Interface
NOx	Nitrogen Oxides
O3	Ozone
OT	Operational Tools

PAS	Port Activity Scenario
PEI	Port Environmental Index
PIXEL	Port IoT for Environmental Leverage
PMS	Port Management System
PM10	Particulate Matter 10 micrometres or less in diameter
PNG	Portable Network Graphics
PV	Photovoltaic
TDS	Total Dissolved Solids
ThPA	Thessaloniki Port Authority
RR	Reliability Ratings
SEA	Significant Environmental Aspects
SO2	Sulfur Dioxide
SDAG	Stazioni Doganali Autoportuali Gorizia S.p.A.
VM	Virtual Machine
XLS	Excel Binary File Format

1. About this document

The scope of the deliverable is to report about the platform integration in ports' infrastructure and the results of the deployment of the use-cases. This is the second iteration of the Integration Report deliverable, which focuses on how the PIXEL solution has been installed, run, validated and functionally deployed in the four ports of the project. All relevant integration aspects (including those organizational and operational) have been described. First, the platform integration is documented. Secondly, this deliverable presents the results of pilots implementations carried out in the different tasks of the WP7, except for the task 7.6 “*Inter-pilot integration and collaboration*” that is described in the D7.3.

1.1. Deliverable context

Table 1. Deliverable context

Keywords	Description
Objectives	<p><u>Objective 1:</u> <i>Enable the IoT-based connection of port resources, transport agents and city sensor networks.</i></p> <p>Definition of validation methodology to be applied for the technical developments of the projects including the installation of the IoT platform in the premises of pilot ports.</p> <p><u>Objective 2:</u> <i>Achieve an automatic aggregation, homogenization and semantic annotation of multi-source heterogeneous data from different internal and external actors.</i></p> <p>Present the integrated version of the PIXEL platform based on the components developed in WP6. Explanation of the validation methodology and the agreement-between-teams step to ensure semantic alignment.</p> <p><u>Objective 6:</u> <i>Develop a methodology for quantifying, validating, interpreting and integrating all environmental impacts of port activities into a single metric called the Port Environmental Index (PEI)</i></p> <p>D7.2 reports the results of the task T7.5, that has had as its goal to deploy the PEI model in all 4 PIXEL ports, including data gathering, agents creation, model installation and provision of results.</p> <p><u>O3, O4, O5 and O7</u> are also linked to this deliverable as this piece reports on the deployment of all models and predictive algorithms in the PIXEL pilot ports, thus demonstrating the achievement of those objectives as well.</p>
Exploitable results	<p>Although all KERs of the project has been isolatedly generated in previous works (WPs and deliverables), D7.2 reports of their integration and usage in real ports, therefore this deliverable is intimately related with the fact of becoming actual exploitable results and not theoretical experiments.</p>
Work plan	<p>This deliverable reports about the works performed in task 7.1, 7.2, 7.3, 7.4 and 7.5. The results reported in this document are based on the developments of the WP4, WP5 and WP6.</p>
Milestones	<p>MS9. The completion of this deliverable (together with D7.3) will mean the achievement of MS9.</p>

Deliverables	<p>Detected inputs:</p> <ul style="list-style-type: none"> • D6.1 and D6.2 describe the PIXEL architecture to be validated • D4.2 and D4.3. Models are fully described with their requirements for integration in D4.2. Moreover, predictive algorithms are being developed and a first description is available in D4.3. • D5.3: The PEI (as a model) is describe and the software to be used in T7.5 is delivered. • D3.2, D3.3 and D3.4 describe the requirements of the PIXEL architecture and the pilots to be developed and evaluated in the WP7. • D7.1 is the first version of the integration report <p>Detected outputs:</p> <ul style="list-style-type: none"> • D7.3 will complement the D7.2, explaining the lessons learnt during the execution WP7 and reporting the rest of WP7 tasks. • D8.3 and D8.4. The pilots developed in WP7 will be evaluated in WP8, both from a pure technical way and from business, economic and social perspective.
Risks	<p><u>Risk N°18</u> - Delay in technical developments. A substantial delay in the completion of the technical developments of the platform and models implies a delay in the validation of the developments and in the development of the different pilot. <i>Mitigation Measures:</i> Firstly, conducting weekly meetings to have a better control of the state of the developments and minimize time deviations. Secondly, making incremental deliveries of software artifacts that allow pilots to advance without waiting for the latest version</p> <p><u>Risk N°19</u> - Technical documentation delayed or incomplete. A delay in documentation or poor-quality documentation will force both technical partners and ports to devote more effort to the development of pilots. <i>Mitigation Measures:</i> Involve the technical leaders of the pilots in the technical follow-up meetings so that they know first-hand the state of the developments</p> <p><u>Risk 20</u> - Delay in the availability of the infrastructure in the pilots. If the infrastructure is not on time, the start of the technical part of the pilot will be delayed and the data and models cannot start to be integrated. <i>Mitigation Measures:</i> be flexible in the minimum requirements so that they have no problems in providing the minimum infrastructure</p> <p><u>Risk 22</u> - insufficient or untimely pilot data available. The data required to develop the scenarios for the different ports are not on time or the quality of the data is not sufficient. <i>Mitigation Measures:</i> work with the ports in the definition of data sources and in the acquisition of the sensors, months before the beginning of the implementation of the pilots. Moreover, it could be possible to test the scenarios with simulated data or look for alternative data sources</p>

1.2. Rationale behind the structure

The deliverable has been structured in eight sections.

- The “PIXEL Validation Methodology” section describes the validation methodology developed and followed in PIXEL to validate the progress of the technical components/tasks of the PIXEL platform and pilots.

- The “Platform integration” section describes the technical and management activities carried out in the project, in order to integrate all the technical components developed in the technical work packages to build the PIXEL platform and the integration mechanisms defined.
- The “Energy Management Trial” describes the pilot developed at the Port of Bordeaux.
- The “Intermodal Transport Trial” describes the pilot developed at the Port of Monfalcone
- The “Port City integration Trial” describes the pilot developed at the Port of Thessaloniki
- The “Port City integration Trial” describes the pilot developed at the Port of Piraeus
- The “Transversal trial” describes the Port Environmental Index development and the results gathered after applying the PEI in the four PIXEL ports.
- The “Conclusion” section summarises the deliverable with the most important conclusions and objectives to be achieved at the end of this work package.

2. PIXEL Validation Methodology

The initial deliverable of this Work Package, D7.1, included a series of provisions for conducting the different pilots in the context of T7.2, T7.3, T7.4 and T7.5.

Within that document (that can be found [here](#)), the team made an exercise for defining the modules of the platform that should be deployed in each port (see 3.2.5, 3.3.5, 3.4.5 and 3.5.5) and how, the models that were required to be integrated and evaluated and the features expected in each of them, thus aligning WP7 work with the design phase of the project (findable in deliverables D3.2 and D3.4). In addition, deliverable D7.1 also outlined a plan for **testing those pilots** in terms of the technical, operational and organizational dimension (see section 3.1.3.). Finally, a set of tools and indications for proceedings with such tests in further stages of the project were also included (see Section 4).

However, there was a crucial aspect that was not tackled by then. This does not respond to a mistake, but rather to a strategical decision. That aspect was: *“how to track that one specific model (in a specific use-case) has effectively been achieved?”*. In other words, the **validation methodology**. The reason why it was not defined in D7.1 is because it does not have a trivial answer. As a matter of fact, this validation methodology required a deep knowledge on the interactions between the modules of the platform, the potential integration mistakes and misalignments that would be appearing and also the various cascade triggers and issues that use to happen in large-scale software deployment actions. By the time of D7.1 preparation (M18), that information was unknown, therefore the team decided to wait till all (modules) development was finished to proceed with the definition of such a methodology.

On month M24-M25, when the isolated developments of modules and models were finished (WP4 ended on M24, T5.3 on M24 and WP6 ended on M26), this validation methodology was closed. This does not mean that till then WP7 tasks were on an “idle state”; on the contrary, they were advancing in the analysis of the software pieces to be put together to be in the best position to define the validation methodology.

While the validation process is a rather internal document (allowing all teams to be aligned and facilitating tracking and control), it has been considered worthwhile to be included in this deliverable. The motivation for this inclusion is based on the following:

- It clarifies the quantity, quality, detail of work that has needed to be performed by PIXEL team in WP7 to achieve full integration of modules and models in the pilots.
- It serves as the baseline for the future “deployment documentation”.
- It can be useful for any external developer/entity aiming at using PIXEL for their use-case, being the first step of a guidelines/instructions in an end-to-end fashion, including operative logic.

In the following sub-sections, the validation methodology is explained:

2.1. Global assumptions and flow

Assumption 1: The validation in PIXEL is done per model/component and per action/task being part of the process (called “block”).

Assumption 2: The validation in PIXEL is based on a set of incremental states for each **component and block**, until the component is completely validated.



Figure 1. Global assumptions in PIXEL model's validation methodology

The following figure shows the global flow of the deployment/integration that has been carried out for the development of the different pilots. This can also be understood as the usual flow that an external port will need to follow for deploying PIXEL and performing a certain use-case.

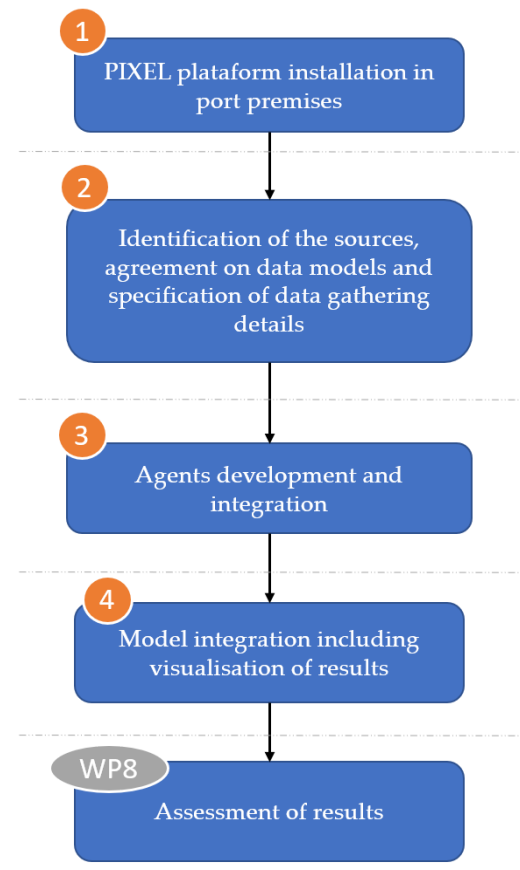


Figure 2. Global flow for the use-case deployment and validation methodology using PIXEL

Just as a side note, a total of 16 models have been “integrated and deployed” in PIXEL ports at the end of the WP7 (5 in GPMB, 3 in ASPM, 4 in THPA and 4 in PPA). That means that this methodology has been followed for 16 items that have been successfully included in ports’ PIXEL instance.

In the following sections, the validation methodology is broken down as follows: per each of the “blocks” (blue blocks on Figure 2), a clear specification of the “incremental states” and the different controls that must be done are detailed.

2.2. PIXEL platform installation block

These were the “incremental states” for validating the installation of the PIXEL platform in the different pilots. As previously mentioned before, these guidelines are suggested to be followed (for WP8 and beyond) to everyone wishing at deploy the PIXEL platform.



Figure 3. PIXEL platform installation block flow of states

- **State 0: Installation**

This state consisted in installing the whole PIXEL platform. For doing so, an Installation Guide has been created, that should be followed. The whole process is boosted by the usage of Docker compose files and Docker containers. Once all components were successfully installed, this step was considered finalised.

To facilitate the deployment and the update of the platform on each port and in order to guarantee that the same components are deployed with the same version on each port, it was decided to rely on GIT¹ and docker-compose to manage the deployment.

A summary of the Installation Guide can be found in the following block (the complete description of the installation process is documented in D6.4 chapter 4.3):

1. Install requirements : docker, docker-compose, git

2. Clone the installation repository

3. Fill the .env and secrets/* files

4. run build.sh (docker-compose build)

5. run install.sh (docker-compose install)

All the files and scripts needed for the deployment are stored on GIT: •
<https://gitpixel.satrdlab.upv.es/marc.despland/Installation>

- **State 1. Validation - Integration between DAL-IH**

This state was included in the reporting validation methodology to represent the “moment” of installation and deployment of NGSI agents (for sensor data and other sources not related with sensors, for instance vessel calls) and the proper transfer of their associated data from the Context Broker (ORION) to the Information Hub.

- **State 2. Validation - Integration between OT-IH**

For installing a model in PIXEL, it is necessary to use the Operational Tools (OT). After the installation, in the execution of the model, it was deemed as necessary to consider the model characteristics and the data sources available. This state aimed at reflecting that “moment” towards achieving the platform installation. OTs and IH must be communicating properly for the whole platform to work. Otherwise, no effective will be noticed after the creation of models.

The acceptance “check” item for moving from this state on, is to analyse whether the execution of a model is properly stored in the IH of that PIXEL instance. In the case of PIXEL, both OT and IH have included a REST API that can be used for that purpose as well.

- **State 3. Validation - Integration between IH-OT-Dashboard**

An IoT platform is not complete (actually, this is a very important part) till the results of the processing of the data are visualised and offered to the end user. For validating such an achievement when installing PIXEL platform, the action required is to create some visualizations for test the correct functioning of the models. To do this task, previously is necessary that the model to test will be able to generate data that could be read by the visualizations.

- **State 4. Validation - Integration between DAL-IH-Dashboard**

Same as for the results of an execution (application over the data), it is extremely relevant for an IoT platform to also visually represent the raw data that was used in that application (as long as those data were stored). For validating the proper integration of DAL-IH-Dashboard, the map that shows a list and the locations of sensors can be checked. Additionally, it was highly recommended to make use of the “View” creator developed in PIXEL named “Custom” that allows a user of the platform to obtain (in the form a Dashboard widget) the last record of any data source as a table.

¹ GIT is a software control version <https://git-scm.com/>

- **State 5. Installation validated**

This state was included in the methodology to consider this “action” (PIXEL platform installation) finalised. It was (and it is and it will be) only reachable after correctly completing the previous states. Once the previous states are correctly installed and validates, the installation of the platform can be considered “finalised”.

2.3.Data integration design block

This stage on the “validation methodology” could actually be considered as a **logical centrepiece** for the deployment of any functionality over the PIXEL platform.

It was created with the purpose of reflecting the need of reaching agreements between the technical team developing the NGSI agents (from raw to structured data) and the developers/owners of a functionality (e.g., a team that has developed a functionality to predict the traffic of vehicles) -which are aware of which data is needed and in which format - to guarantee a solid understanding basis to minimise inefficiencies and allow a proper deployment of services over PIXEL platform.

In that regard (and with the intention to help future users of PIXEL), the agreements that were needed to be taken between both teams were:

- Establishment of a clear list of data needed to run the model.
- Clarification of the periodicity required of that data (per second, per hour, per day, per week, etc.).
- The data model that must be used by the agent to send to the Context Broker (as always, FIWARE data models recommended – check with Orange). The next step after identifying the need of the data was (is and will be) to know how to obtain this data from a sensor or an Information system and to define the semantics of the data. In PIXEL, the semantics is defined by data models (<https://www.fiware.org/developers/data-models/>). If there is not a proper DataModel fitting the data needs, PIXEL members defined their own, custom data model. These Data Models will be pushed to the FIWARE Community after validation of their use, in the PIXEL pilots. All the data models used in PIXEL are stored in a centralised repository: https://gitpixel.satrdlab.upv.es/iglaub/Data_Models. More information about Data Models in PIXEL can be found at **Appendix A**.
- The name of ElasticSearch-IH index where the persistent data must be stored. As per team decision in PIXEL, when the data comes from an IoT source (i.e., is not a result of a PIXEL process/model), the index starts by: “arh-lts-...”.
- Thresholds, alarms and typical values both for plain data and for the results of the model.
- Clarification about the GetInfo and Instance JSON files of each model and predictive algorithm to be aligned with the OT framework (see WP6 deliverables).

With that information agreed (again, per model that the deployment aims at incorporating), both teams started (/will be able to start) working separately: on one side the agents developers and on the other the model owners, dockerising, integrating them and setting up the layout and visualisations within the PIXEL platform functioning.

2.4.NGSI agents validation block

This section explains how to validate the integration of a data source into the PIXEL platform. Same as the two previous, it aims at representing the flow that was followed in all models in every PIXEL pilot and it also aims at creating a replicable set of guidelines for future users. To integrate a data source into the PIXEL platform, a set of steps should be followed, ranging from the identification of the need of a specific data until the data is stored in the PIXEL platform and later, it is used by models or by the visualizations of the dashboard.

However, before digging deeper into the flow, for the sake of understandability of the terms of the sub-section, some context below on NGSI agents in PIXEL:

The purpose of NGSI Agents is to connect external data sources to the FIWARE Context Broker (ORION) through the NGSI API. Usually, those agents are built for IoT, but they could be used with any kind of data source.

There are two types of agents:

- Agent that offers an API to allow the source to push new data
- Agent that connects to the source periodically to retrieve information

In order to do its job, the agents are divided in three mains blocks:

- The source connector that exchanges the data with the data source
- The data transformer that transforms the data from the source format to NGSI compatible format
- The NGSI connector that exchanges the data with the FIWARE Context Broker

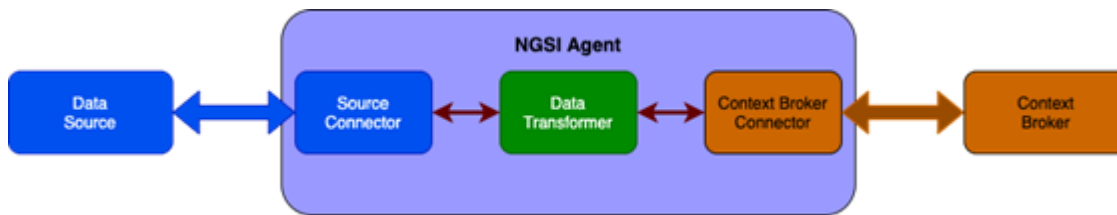


Figure 4. PIXEL NGSI agents breakdown

The flow in the NGSI agents validation block within the whole validation methodology is:



Figure 5. PIXEL NGSI agents validation block flow of states

- **State 0: Data available in the port and accessible by the partners**

The initial state that was included was to ensure having the data available. It is worth to mention that this achievement is not trivial (see the HOPU acquisition below). As it has been reported in many deliverables of PIXEL, data acquisition should be considered as a first-class issue to tackle in any PIXEL deployment. In some cases, it was needed to acquire a device or sensor to generate this information which had to be selected, bought and installed. In other cases, the data was provided by a third-party service. In any case, the suitability of the data sources for the purpose for which they are to be used must be considered.

In that sense, a remarkable effort needed to be done in order to “have available” the data produced by the sensor purchased by APT, GPMB and THPA to measure noise and light in real time. The market analysis related to new sensors to be acquired was already described thoroughly in deliverable D5.3, so it is not the objective here to relate on that. Nonetheless, although the device was specifically selected due to its “closeness” to FIWARE-friendly IoT projects (among others), the integration required special work to be devoted by PIXEL partners. The process followed has been documented and it is included in this document as an annex in **Appendix B**.

- **State 1: NGSI Agent created**

Once the previous state has been secured, the NGSI agent can start to develop. In the case of PIXEL, a specific Python framework (own library that now is public via PyPI) was created with the purpose of simplifying the development of the agents (<https://gitpixel.satrdlab.upv.es/orange/pyngsi> or [-public-https://github.com/Orange-OpenSource/pyngsi](https://github.com/Orange-OpenSource/pyngsi)). The current version of this library supports natively three different types of data sources: (1) Expose an API to collect data in CSV or JSON format, (2) Collect data from source providing JSON or CSV from an API or FTP and (3) Subscribe to MQTT channel. The utilisation of pyngsi for other type of data sources is possible but a bit of code is needed to be developed. As a matter of fact, the latter has been the predominant case in PIXEL.

Arrived at this point, the action required was to effectively develop the NGSI agent. Even though pyngsi facilitates the task, completing an agent is a time-consuming, technically challenging activity. Besides, it differs

from one case to the other thus replicability is possible but not immediate. To put this action into context, more than 25 agents have needed to be developed in PIXEL to meet the functional objectives of the project.

Therefore, apart from the commodity library, a full-fledged manual on the development of NGSI agents (contextualised for PIXEL) was created. The so-called NGSI Agents cookbook can be found in **Appendix C** at the end of this document. These guidelines were used all along WP7 to deploy the needed agents in all PIXEL ports.

After the development, to validate the proper functioning of the agent, a set of unit tests should be executed. If the execution ends without errors, the results obtained must be observed and followed to check the correct operation of the agent. All PIXEL agents deployed in each port successfully ran those tests and thus were considered ready to be tested locally and afterwards deployed in ports' premises. This step is recommended to be done (to external users) with the aim of not incurring on any inefficiencies.

- **State 2: NGSI Agent Tested locally**

Once the agent had been developed and validated in an isolated way, it was time to validate its integration in the platform. According to the plan defined (and also after the experience of the deployment of the agents), this validation is recommended to be decoupled in three sub-steps (these must be executed in the local environment):

- **Agent up and running:**

The first step is installing (NGSI agents in PIXEL are managed as Docker containers built from images available on the Git Repository of the project) and check if the code does the work for which has been developed and it does not produce any kind of error. For accomplishing this validation, a set of integration tests should be executed.

- **Data received by the Context Broker (ORION)**

If the installation is correctly, the next step is check if data sent by the agent and received by the Context Broker had been correctly processed.

- **Data stored into the IH:**

Once data received by the Context Broker is checked, this data will be stored in to the IH. The third sub-step to check proper local deployment of the agent is to check whether the information that had been updated in the Context Broker (entity) is properly persisted in the long-term storage database system (new entries in a document within an index).

- **State 3: NGSI Agent Tested in the Development environment**

The procedure for validating this “state” of the block is equivalent to the previous state. However, the only difference is that the sub-steps must be executed in the development environment (“INT” instance – for tests and development).

- **State 4: NGSI Agent Tested in at least one Pilot environment**

The procedure of this state is the same as in the two previous states. However, difference between this step and the others two is that the sub-steps must be executed in the pilot environment. This is how it was done in all 4 PIXEL pilot environments

- **State 5: NGSI validated and ready to be used by models and the Dashboard**

This state was included in the methodology to consider this “action” (NGSI agents validation) finalised. This state is only reachable after the data passes all relevant validations. If the data is correctly processed it can be used by models and dashboard.

2.5.Models/predictive algorithms validation block

This section explains how to validate a model into the PIXEL platform. This last block is the one considered in this methodology as worthwhile to be broken down into several states. Here is why: in order to validate a model from its development until its integration into the PIXEL platform, it should follow a set of incremental validation steps from the development of the model until the model is executed in the PIXEL platform.

- **State 0: Model requirements defined (drawing from the results of the agreements between model owners and agents responsible)**

This initial step consisted in defining the requirements that the model must meet, and since it is a precondition, this step should not require any development action. As soon as this is accomplished, the model should develop and begin to integrate. Again, it is highly recommended to potential adopters to pay enough attention to this matter.

- **State 1: Model developed**

This validation methodology assumes that a model has already been developed, tested in an isolated way (namely using past data and not integrated in PIXEL).

In PIXEL project, that was the case. Various models and predictive algorithms (D4.1 and D4.3) were developed in different programming languages, and for their validation the models were executed in a locally and isolated environment using an example of real data (in WP4).

- **State 2: Model Dockerized and Tested locally (local platform instance)**

Once the model has been developed and validated in an isolated way, it was time to validate its integration in the platform in a local environment. To proceed with the validation of this step, it is needed to have the model dockerized. This validation is composed of four sub-steps:

- **Prerequisites:** The first step is installing and check if the code does the work for which has been developed correctly and not produce any kind of error. If the agents are not available, at this state, it is possible to bypass this issue and insert the data manually into the Elasticsearch.
- **Model installation:** After insert data, either automatically by the NGSI agent or manually, the model is considered installed.
- **Store data:** The next sub-step is executing the model (see scheduling through Operational Tools in D6.5) and checks that the output data is correct stored in the IH without error or data loses.
- **Create visualizations:** Create several visualizations using the output produced by model executions and verify that the visualizations are showing the expected data in the dashboard. This was done in all PIXEL pilot trials.

- **State 3: Model Tested in the development environment**

The procedure of this state is the same as in the previous state. However, there are two differences between states. The first one is that it is necessary to use NGSI agents to insert data into the IH, since this insertion of data cannot be done manually. And the second difference is that the sub-steps must be executed in the development environment.

- **State 4: Model Tested in at least one Pilot environment**

The procedure of this state is the same as in the second state. However, as in the previous state, there are, two differences between this state and the second. The first one is that it is necessary to use real NGSI agents and real data to insert data into the IH. And the second one is that the sub-steps must be executed in the pilot environment.

- **State 5: Model validated**

When the model reaches this step, it means that it has passed all the relevant validations and can be used in the dashboard.

3. Testing PIXEL deployments

The objective of this section is to report how has been the “transition” from the pure platform installation to the actual functioning provision to ports’ stakeholders, which in PIXEL has been overall called “testing”.

While the previous section was focused on reporting the “operational flow” for ensuring (and validating) a proper PIXEL deployment, the sub-sections below report on the technological actions undertaken to ensure that such a validation will make sense and allow to meet pilot’s objectives.

First (before the deployment of actual pilots), it was decided to envision and “put in practice” a small scale, “functionally dummy” end-to-end scenario. The objective was to both: (i) get comfortable with the instructions and validation procedures, and (ii) calibrate such a methodology, if needed. In order to test that whole end-to-end scenario integration there was a need to use a basic model and follow the validation methodology explained in Section 2.

Second, following the provisions in the Integration Plan (see D7.1), a series of tests related to the use-case stories were performed in order to actually validate the pilots and complete the trials (a trial is, at the end of the day, a functional testing of a development in a targeted vertical).

The outcome of the latter work has been, as a matter of fact, the completion of all tasks in WP7. Meaning: providing to the ports a functioning platform with the functionalities requested (based on D3.4) that can be accessed through a web portal and that is ready for technical and business/economic assessment.

3.1. End-to-end scenario testing

This sub-section aims at documenting the results/conclusions extracted from the “end-to-end scenario testing” action that took place in WP7. This scenario basically implied:

- A basic agent sending data to the system, similar to a ‘ping’ command. This part allowed to test (i) the deployment of agents in the platform, and (ii) the monitoring of the data flow from the Context Broker to the Information Hub (IH).
- A basic model, able to process the incoming data and show a result (e.g., count pings within a time frame). This part allowed (i) to test the dashboard and operational tools to publish the model, (ii) to test the dashboard and operational tools to run and schedule the model, (iii) to test whether the model is able to retrieve data from the IH, (iv) to test whether the model is able to store data in the IH, and (v) to test displaying a visualization in the dashboard to show the result.

The deployment of the scenario followed the validation methodology exposed in Section 2, while the different relevant conclusions are structured in a step-by-step list that can be found at Appendix D.

3.2. Formal Testing

Once the “ping scenario” was validated, all pilots used the validation methodology and performed the actions to achieve their objectives.

In parallel (although mostly at the end of the period), official testing took place. This testing was based on the indications in D7.1 and have been performed using the tool TestLink (<http://pixeltesting.prodevelop.es/testlink/login.php>). This testing was composed of:

- Unit tests, module tests and integration tests on every technological module of the platform.
- A series of test cases defined for the different user-stories and requirements of the pilots. All of them were created in TestLink and have been tracked using the tool.

The reports of TestLink will be available after WP8 ends, as it is the intention of the team to keep refining the test cases and to be adjusted to the analyses on T8.2.

4. Energy management trial – Port of Bordeaux

4.1. Context review

As defined in D7.1, GPMB use-case aims to use PIXEL to:

- Understand the energy needs and the impact of the different port activities in terms of energy consumption.
- Be able to optimize and well dimension renewable energy networks.

These two questions have been addressed one after the other. Firstly, the PAS model has been developed and integrated in the GPMB platform to model the different port activities. In parallel, PAS forms have been defined to offer a user-friendly interface for defining machines and supply chains. Secondly, the outputs of the PAS model have been used by the energy model to give instant and cumulative results, about the energy used by the different port activities. Finally, the results about the energy used during port activities have been compared with the potential of energy production using a PV system (based on the work done in WP4).

In deliverable D3.4 “Use cases and scenario manual V2”, GPMB has described eight different scenarios to test and integrate with the PIXEL platform:

- Statistics manager scenario (GPMB-StM-1) which aims to provide a data analysis of vessels’ calls, by connecting VIGIESip to PIXEL Platform, to obtain historical data and then use small analytics tools. This has been covered by the integration in the IH of PIXEL of all the data related with vessel calls (planned and historical) coming from the PMS of GPMB. Through the PIXEL dashboard vessels calls can be visualised.
- Energy manager scenario (GPMB-EM-1) which aims to understand the energy consumption of the port, based on vessels’ calls and supply chains description. To do so, the port activity scenario model and the energy models developed in WP4, have been integrated, parametrized and used.
- Energy manager scenario (GPMB-EM-2) which aims to connect smart electrical sensors to the PIXEL platform. A specific agent has been developed to retrieve this data. The agent is ready to be used, as soon as GPMB can create the IT access to the data.
- IT manager scenario (GPMB-IT-1) which aims to connect old sensors of the port to the PIXEL platform. Specific agent has been developed to gather data for old sensors, like the tide level sensors. The associated data are stored in PIXEL IH and can be visualized using the PIXEL dashboard.
- Environmental manager scenario (GPMB-EnvM-1) which aims to use PIXEL and PEI as a support for environmental decisions. The PEI model and the associated agent have been developed and integrated in GPMB.
- Port manager scenario (GPMB-PM-1) which aims to use PIXEL as a support for investment in green energy. The PIXEL platform installed in GPMB provided results and visualisation to compare production (using a PV system) and consumption of energy (using the outputs of the energy model).
- Software editor scenario (GPMB-SE-1) which refers to the integration of the PIXEL platform in VIGIESip (GPMB’s PMS). The PIXEL platform provides a different API that allows the PMS of GPMB to directly retrieve data stored in the PIXEL Information Hub.

Port Agent scenario (GPMB-PA-1) which refers to transferring or extending some of the previous functionalities/models to the port ecosystem (e.g., terminal operator). PAS model, energy model and PEI can be used by different port agents. The PIXEL dashboard allows to define permissions and roles and thus make possible to share results and visualisation of data with the port ecosystem.

4.2. Platform infrastructure

The deployed IT infrastructure is hosted by a third party to facilitate interventions. It is composed of two servers:

- Server 1: named PIXEL PUBLIC
 - 4vCPU, 16Go and 150 Go HDD
 - OS Ubuntu Bionic
- Server 2: named PIXEL CORE
 - 8cCPU, 32 Go, 300 Go HDD
 - OS Ubuntu Bionic

4.3. Data sources

In this section, all data sources integrated into the Pilot of Bordeaux (GPMB) – corresponding to task T7.2- are described. The integration strategy used in every case depends on the data source.

4.3.1. Vessel calls (planned)

Table 2. Data source “Vessel calls planned” Port of Bordeaux

Data Name	Vessels calls planned
Data Description	VesselCall NGSI is a Python NGSI agent that listens for vessel-call logs of VIGIESIP. It lists the vessels calls that plan to load or unload in GPMB
Data Origin	VIGIESIP (Port Management System of GPMB)
Data Format	JSON
Integration	NGSI Agent
Agent Name	vcall-ngsi https://gitpixel.satrdlab.upv.es/orange/vcall-ngsi
Data model	VesselCall https://gitpixel.satrdlab.upv.es/marc.despland/DataModels/src/master/DataModels/VesselCall.md
Update frequency	Daily
Used by	PAS model, Energy model, PEI, ETD predictive algorithms

4.3.2. Vessel calls (historic)

Table 3. Data source “Vessel calls history” Port of Bordeaux

Data Name	Vessels calls history
Data Description	VesselCall NGSI is a Python NGSI agent that listens for vessel-call logs of VIGIESIP. Historical data of all the vessels calls after customs' treatment

Data Origin	VIGIESIP (Port Management System of GPMB)
Data Format	JSON
Integration	NGSI Agent
Agent Name	vcall-ngsi https://gitpixel.satrdlab.upv.es/orange/vcall-ngsi
Data model	VesselCall https://gitpixel.satrdlab.upv.es/marc.despland/DataModels/src/master/DataModels/VesselCall.md
Update frequency	Daily
Used by	PAS model, Energy model, PEI, ETD predictive algorithms

4.3.3. Tide level

Table 4. Data source “Tide Level” Port of Bordeaux

Data Name	Tide Level
Data Description	Network of tide level sensors across the Gironde estuary
Data Origin	Tide level platform (NAMI)
Data Format	JSON
Integration	NGSI Agent
Agent Name	Frbodtidesensor https://gitpixel.satrdlab.upv.es/marc.despland/frbodtidesensor
Data model	TideSensorObserved https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/TideSensorObserved
Update frequency	Hourly

4.3.4. Air pollution ATMO

Table 5. Data source “Air Pollution ATMO” Port of Bordeaux

Data Name	Air Pollution ATMO
Data Description	Atmo NGSI is a Python NGSI agent that requests air quality measures from ATMO Nouvelle Aquitaine.
Data Origin	ATMO API
Data Format	JSON
Integration	NGSI Agent
Agent Name	Atmongsi https://gitpixel.satrdlab.upv.es/orange/atmo-ngsi
Data model	AirQualityObserved https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/AirQualityObserved
Update frequency	Daily

4.3.5. Weather station

Table 6. Data source “Weather Station” Port of Bordeaux

Data Name	Weather Station
Data Description	Sencrop NGSI is a Python NGSI agent that requests a Sencrop weather station installed in GPMB
Data Origin	Sencrop weather station
Data Format	Daily
Integration	NGSI Agent
Agent Name	WeatherObservedSencrop https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/WeatherObservedSencrop

Data model	WeatherObservedSencrop https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/WeatherObservedSencrop
Update frequency	Daily

4.3.6. Noise/PM10/Light

Table 7. Data source “Noise/PM10/Light” Port of Bordeaux

Data Name	Noise/PM10/Light
Data Description	Hopu NGSI is a Python NGSI agent that receives messages published on the MQTT bus by HOP Ubiquitous Smart Spot devices
Data Origin	HOP Ubiquitous Smart Spot devices
Data Format	[format of the data: .csv, json, plain test]
Integration	NGSI Agent
Agent Name	hopu-ngsi https://gitpixel.satrdlab.upv.es/orange/hopu-ngsi
Data model	Hopu https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/Hopu
Update frequency	Daily
Used by	PEI

4.3.7. MARPOL Annexes – Data of ships

Table 8. Data source “MARPOL Data Ships” Port of Bordeaux

Data Name	MARPOL Data Ships
Data Description	The MARPOL NGSI Agent collects maritime pollution data.
Data Origin	VIGIESIP (Port Management System of GPMB)
Data Format	JSON

Integration	NGSI Agent
Agent Name	marpol-ngsi https://gitpixel.satrdlab.upv.es/orange/marpol-ngsi
Data model	MarpolWaste https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/MarpolWaste
Update frequency	Daily
Used by	PEI

4.3.8. Waste terminal – Terminal of Bassens

Table 9. Data source “Waste terminal Bassens” Port of Bordeaux

Data Name	Waste terminal Bassens
Data Description	The marpol-terminal NGSI agent collects data about waste generated by a terminal
Data Origin	Excel files
Data Format	Excel files
Integration	NGSI Agent
Agent Name	marpol-terminal https://gitpixel.satrdlab.upv.es/orange/marpol-terminal-ngsi
Data model	MarpolTerminalWaste https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/MarpolTerminalWaste
Update frequency	Monthly basis
Used by	PEI

4.3.9. Estimation of photovoltaic production per day

Table 10. Data source “Estimation of PV production per day in the port” Port of Bordeaux

Data Name	Estimation of PV production per day in the port
Data Description	According to several inputs described in D4.1 and D4.2 , a 11-year span (2005-2016) range of values were provided in the form of a CSV illustrating the Power generation of PV panels on specific conditions in the Port of Bordeaux. The measurements (rows in the CSV) have a periodicity of 1 hour
Data Origin	PVGIS https://ec.europa.eu/jrc/en/pvgis
Data Format	csv
Integration	NGSI Agent
Agent Name	ngsi-agent-frbod-photovoltaic https://gitpixel.satrdlab.upv.es/iglaub/ngsi-agent-frbod-photovoltaic
Data model	PhotovoltaicMeasurement https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/Photovoltaic_Measurement
Update frequency	Hourly basis
Used by	Energy Model

4.3.10. Port parameters

Table 11. Data source “Port parameters” Port of Bordeaux

Data Name	Port parameters
Data Description	All data related with the port activities description: machine, energy use, supply chain, ...
Data Origin	PAS forms completed by GPMB
Data Format	JSON
Integration	Forms integrated through the PIXEL dashboard
Update frequency	Defined once and then can be modified by the user

Used by	PAS model, Energy Model
----------------	-------------------------

4.4. Functionalities provided, models and algorithms

The table below summarises the functionalities that have been made available to the port in the trial. The fact of having deployed those functionalities (based on that series of agents and not others) directly maps to the needs expressed in the use-case stories and the pilot objectives (see Context Review above). The table below also refers to the models that have been installed and run using the data fed by the agents described above.

Table 12. Functionalities deployed in GPMB – task T7.2






Functionality	Model	Inputs	Results shown
Energy consumption - temporal line	PAS model	Vessel calls (planned), vessel calls (historic), port parameters	A temporal line (evolution graph) with the energy consumed, separated per type of fuel (diesel, electricity...)
Terminal planning (Gantt)			Visualisation in the form of a Gantt distribution the arrival at the port and processing in the terminal associated to every vessel.
Renewable balance			Comparison (in two lines in a same graph) of the temporal evolution of the energy demanded for operating the vessels in the port (result of the PAS) and the energy that would be generated in that period of time by eventual PV panels.
Estimated Time of Departure prediction	ETD Predictive Algorithm	Vessel calls (planned), Vessel calls (historic)	Estimated time of departure of each vessel that is operated in the port. The visualisation is similar to the Gantt.
PEI	PEI model	eKPIs	(reported in section 8 below)

In the following sub-sections, there are some evidences and reflections over the results of the deployment of these functionalities. In the scope of WP7, the successful deployment of all the models in the table (= providing satisfactory, reasonable results) was a sufficient condition to indicate the T7.2 task as finalised. However, this “handover” of running models in the platform was complemented with an explanation of the usage of each model and the interpretation of the results. This “handover” and explanation took place on the Plenary Meeting held virtually at the beginning of June 2021.

The way that those evidences are organised in the sub-sections below is **per model** (2nd column of table above). PEI is reported in Section 8 of the document.

4.4.1. PAS model

Table 13. Model “PAS Model” Port of Bordeaux

Model name	PAS model
Model Description	<p>PAS has been developed integrating constraints and needs of the small and medium ports and with the objective to convert raw data into useful information about “today” and model port activities to predict “future” information. The first step transforms the fed raw data into a Port’s Activities Scenario (PAS) which is a list of all the upcoming port’s operations, with determined articulation across time. The second step calculates the outcome of the PAS on a certain period (as energy consumption, pollutants emission etc.). The PAS model can describe all operations related to a cargo transition that take place within a port.</p> <p>Its purpose is to establish an operational description of the port activities related to cargo handling. The produced PAS is a fulfilled data-model listing all the considered activities’ time series. Because data is a key point to having useful models and since data collection is an intensive phase, the PAS model can be used with different levels of details regarding input data.</p>
Inputs requested	<p>Input data of the PAS model are mainly based on:</p> <ul style="list-style-type: none"> i) vessel planning (arrival date, cargo type and tonnage), ii) activity data (details related to the transfer of cargo), iii) operational data (refer to the technical specifications of engines and equipment used) iv) emission source data (related to emissions factors of engine and equipment).
Outputs generated	<p>The outputs generated are divided in several blocks, which aim at representing the different interesting outcomes related to the operations of a maritime terminal to process a certain amount of cargo:</p> <ul style="list-style-type: none">  areas_occupancy.json  energies_consumptions.json  handlings.json  machines_uses.json  pollutants_emissions.json <p>PAS is capable of (for a specified time range) simulate how much each area of the port will be occupied, how much energy will be consumed (per area, machine and cargo processed) and the emissions that will be generated. The records are associated to timestamps with a granularity of 15 minutes.</p> <p>The knowledge and modelling of the supply chain and port activities (machine type, duration of use, position in the port) enable the building of activity scenarios that are used to identify the energy sources and local emissions of pollutants, but also to estimate the flow of cargo entering or leaving the port. Using this approach, the resulting modelling scenario might be used by the ports as a support for decision making. The Port Activities Scenarios is used to link different models and guarantee interoperability. Depending on the available data via the PIXEL Information Hub, a scenario can be completed with</p>

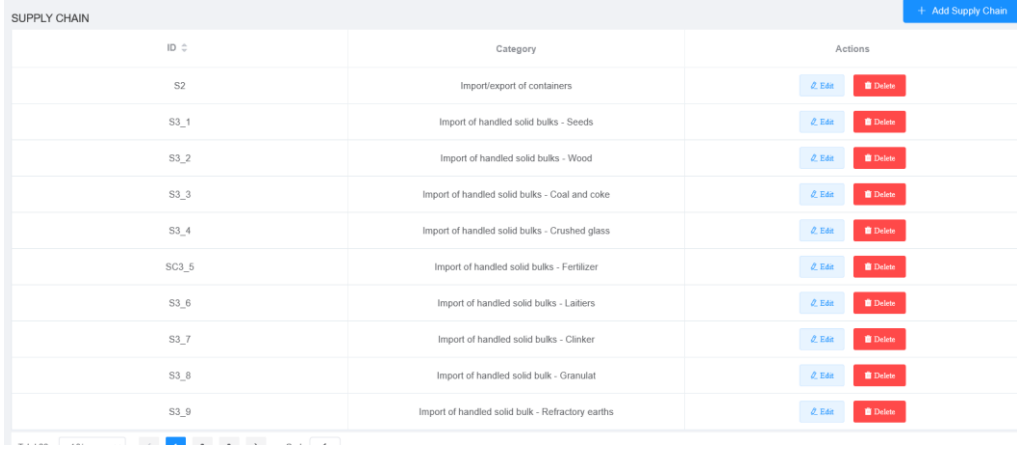
	<p>models related to energy, environmental pollution, and area occupancy. PIXEL proposes a model that enables ports to calculate their energy consumption relating to their activities. The Port Activities Scenario (PAS), combined with the energy model and emission factors, is a transferable and applicable tool for small and medium European ports that allows to model port supply chains. The output of this model can be used as an input for evaluating the energy consumption and with the use of emissions factors to provide an estimation of pollutants emissions. PAS output allows us to know, with a simulation perspective, the total amount of energy consumed by the machinery needed to operate one vessel. Using that information, altogether with the type of fuel/power used, can be leveraged by the PEI, to estimate the air emissions attributable to the terminal activity in a period. This functionality is directly included as a module of the PAS.</p>
Integration	Docker (DockerHub image: erwansimon/pas_model)
Description of the model visualisation = functionality wished for complying with pilot's objectives	<p>PAS forms inputs</p>  <p><i>Figure 6. Port parameters being introduced before running the PAS (sample)</i></p> <p>Gantt visualization</p>



Figure 7. Gantt visualisation of the results of the PAS model

Energy consumption evolution:

This functionality aims to enable ports to calculate their energy consumption related to their activities. Since energy demand at the port is directly proportional to the movement of cargo, special focus is placed on this specific task. The energy demand is modelled according to a fixed scenario using the PAS.

The output here consists of information addition into the PAS. For every operation, the energy consumption is inserted as a new sub-field.

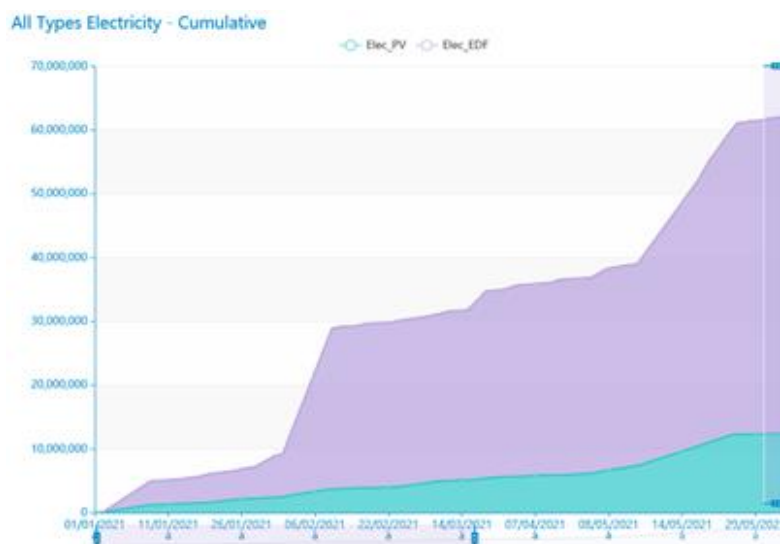


Figure 8. Energy consumption evolution out of PAS model (cumulative)

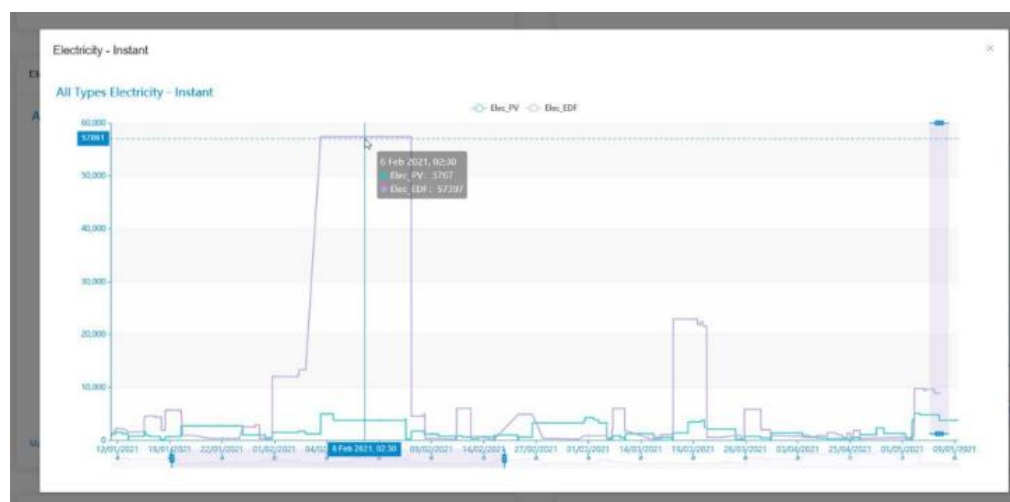


Figure 9. Energy consumption evolution out of PAS model (instant)

Renewable balance

This functionality aims to predict the electrical energy that could be produced by the port using photovoltaic installation, compare it with its electricity consumption and help GPMB to invest in a PV system.

The output of the PVGIS model (used as an input in PIXEL) is a csv file containing the electricity power production as a time series as presented in the table below 5, where *EPV* is the PV system power (W), *G_i* is the in-plane irradiance (W/m²), *As* is the sun elevation (deg.), *Tamb* is the ambient temperature (deg. C), *W10* is the wind speed (m/s) and *Int.* indicate if the solar radiation values are reconstructed (when the value is 1). Here below there is a sample of such information:

Table 14. Sample of PVGIS model output used in the renewable balance functionality

Date	EPV	G _i	As	Tamb	W10	int.
20070101:0055	0.00	0.00	0.00	13.04	3.34	0
20070101:0155	0.00	0.00	0.00	12.70	3.34	0

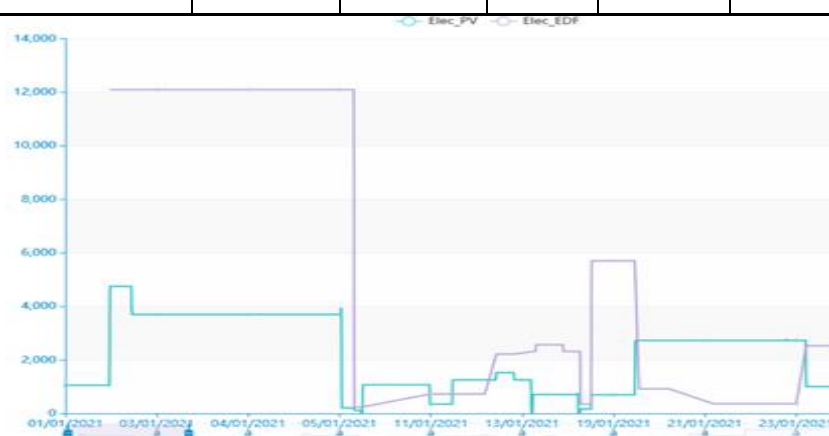
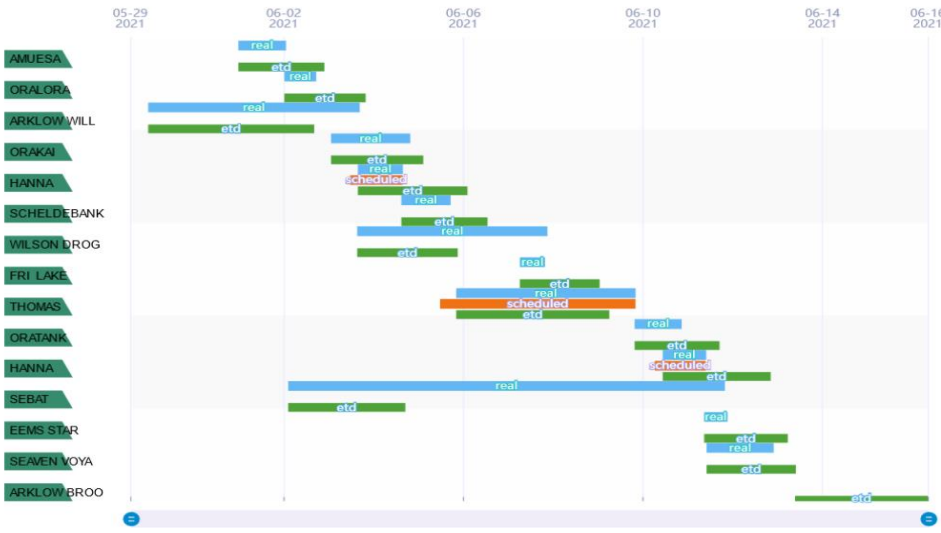


Figure 10. Renewable balance functionality visualisation

4.4.2. ETD Predictive Algorithm

Table 15. Predictive Algorithm “ETD” Port of Bordeaux

Model name	ETD Predictive Algorithm
Model Description	This algorithm predicts the Estimated Time of Departure (ETD) for all the vessels that arrive or are expected to arrive at the port. The input needed corresponds to that included in the documents known as FAL forms. The evaluation of this algorithm was included in Deliverable D4.4.
Inputs requested	Data from the Vessel Calls Agent.
Outputs generated	The Estimated Time of Departure (ETD) for every vessel in the port.
Integration	The algorithm is available as a Docker image that communicates with the Information Hub for data retrieval and storage.
Description of the model visualisations	<p>Gantt visualisation</p>  <p>Figure 11. ETD Predictive Algorithm results visualisation in PIXEL</p>

5. Intermodal Transport trial – Port of Monfalcone

5.1. Context review

As described in detail D7.1, ASPM use-case aims to use PIXEL to:

- Share between the Port of Monfalcone and SDAG to reduce bottlenecks and congestion in port areas and to promote the use of railways or inland terminals where necessary;
- Reinforce the safety related to ADR transport through the operability of data with regional stakeholders and in connection with SILI;
- Collect data on Regional logistic flows to support the activities of the Regional Environment and Health Observatory (REHO). Here the objective is to give the REHO the possibility to monitor the flows on

the road. Within the ARPA (Regional Environmental Agency of Friuli Venezia Giulia), the REHO structure has the aim to put in relation health population status and environmental factors;

- Collect and analyse data on Regional logistic flows to support the activities of the Regional Government (another public entity) to evaluate specific programs to promote rail logistic solutions as well as the implementation/expansion of other logistic infrastructures.

Both Traffic Prediction Algorithm and Intermodal Transport Model are aimed, in particular, at forecasting possible congestions, which could take place at the entrance of the Port and impact the traffic of the city of Monfalcone, with a significant impact in terms of both pollution and safety. Such models, based on near real time data provided by both Vessel Calls and the SILI platform, allow ASPM's management to deal with such events and define backup strategies, such as the usage of SDAG's parking area as a buffer for incoming vehicles. PEI and PAS have been exploited, on the other hand, in order to identify and assess the impact of port related activities on the environment, in terms of both pollution and energy consumption.

In deliverable D3.4 "Use cases and scenario manual V2", ASPM and SDAG have described six different scenarios to test and integrate with the PIXEL platform:

- Gate Manager (PoM-GM-1): aimed at forecasting parking occupancy in the port entry parking area and prevent congestions, by rerouting trucks to SDAG in order to minimize traffic jam;
- Environmental Manager scenario (PoM-EM-1): aimed at collecting and analysing environmental data to both support the REHO and plan future investments/procedures to promote a greener port;
- Software editor scenario (PoM-SE-1): aimed at assessing features required by the new versions of the SILI platform, in order to support port's operations in a proper way;
- Parking area Manager (SDAG-PM-1), Parking area Manager (SDAG-PM-2), Parking area Manager (SDAG-PM-3): aimed at simulating and forecasting traffic (in terms of truck and dangerous transport) directed to SDAG, coming from the regional highway or from the ASPM port area, in response to a specific congestion.

5.2. Platform infrastructure

The infrastructure provided by INSIEL in order to support the setup of the PIXEL platform at ASPM includes 2 virtual machines with following features (for each VM):

- 4 vCore;
- 8 GB of RAM;
- 512 GB of storage;
- Centos 7 Operating System.

The VM A is devoted to the execution of the different containers constituting the PIXEL platform and the related models (including PEI and predictive algorithms). The VM B acts as a proxy to publicly expose a subset of services included in VM A on the Internet.

Following services are publicly available:

- Access Management: <https://id-pixel.insiel.it>
- Data Acquisition Layer: <https://dal-pixel.insiel.it/orion/>
- Information Hub: <https://admin-pixel.insiel.it/ih/proxy>
- Kibana: <https://admin-pixel.insiel.it/kibana/>
- Dashboard and Operational Tools: <https://dashboard-pixel.insiel.it>
- Monitoring: <https://admin-pixel.insiel.it/nagios/>

Access to each service is secured by the access management module or by basic authentication. Servicing and maintenance are provided by authorized users by means of a VPN connection.

NSGI Agents have been deployed on a dedicated server, located outside the perimeter of the INSIEL's server farm, in order to easily overcome limitations related to security policies applied to FTP protocol. FTP protocol, in fact, is required by the different information sources (e.g.: smart cameras, SDAG Access Control System, etc.) to feed the DAL, the IH and, more generally, the models.

5.3. Data sources

In this section, all data sources integrated into the Port of Monfalcone – corresponding to task T7.3- are described. The integration strategy used in each case depends on the data source.

5.3.1. SILI

Table 16. Data source “SILI” Port of Monfalcone

Data Name	SILI (URN of the data source for identification within PIXEL: urn:pixel:DataSource:itmnf:SILI)
Data Description	Number of trucks identified (on an hourly basis) by the cameras constituting the SILI platform, for each relevant spot located on the regional highway network or at ASPM and SDAG entrances.
Data Origin	Data is provided by a batch executed on the database of the SILI platform on a daily basis and sent to the agent via FTP.
Data Format	XLS file
Integration	By means of a dedicated agent
Agent Name	Insiel Agent – SILI process https://gitpixel.satrdlab.upv.es/paolo.casoto/PIXEL_INSIEL/src/master/PIXELInsielAgent
Data model	TrafficFlowObserved https://github.com/smart-data-models/dataModel.Transportation/tree/master/TrafficFlowObserved
Update frequency	Daily
Used by	Intermodal transport model, Traffic Predictive Algorithm

5.3.2. Available spots in the parking of SDAG

Table 17. Data source “SDAG_ACS” Port of Monfalcone

Data Name	SDAG_ACS
Data Description	Number of available parking spots at SDAG.

Data Origin	Data is provided by the STAG's Access Control System platform on an hourly basis and sent to the agent via FTP.
Data Format	CSV file
Integration	By means of a dedicated agent
Agent Name	Insiel Agent – SDAG process https://gitpixel.satrdlab.upv.es/paolo.casoto/PIXEL_INSIEL/src/master/PIXELInsielAgent
Data model	OffStreetParking https://github.com/smart-data-models/dataModel.Parking/tree/master/OffStreetParking
Update frequency	Hourly
Used by	Intermodal model

5.3.3. Smart Camera

Table 18. Data source “Smart Camera” Port of Monfalcone

Data Name	Smart Camera (URN of the data source for identification within PIXEL: urn:pixel:DataSource:itmnf:SmartCamera)
Data Description	Number of available parking spots at ASPM and presence of congestions inside the parking area.
Data Origin	Data is provided by the ASPM's Smart Cameras platform on an hourly basis and sent to the agent via FTP.
Data Format	CSV file
Integration	By means of a dedicated agent
Agent Name	Insiel Agent – ASPM process https://gitpixel.satrdlab.upv.es/paolo.casoto/PIXEL_INSIEL/src/master/PIXELInsielAgent
Data model	OffStreetParking https://github.com/smart-data-models/dataModel.Parking/tree/master/OffStreetParking

Update frequency	Hourly
Used by	Intermodal model

5.3.4. Vessel Call

Table 19. Data source “Vessel Call” Port of Monfalcone

Data Name	Vessel call (URN of the data source for identification within PIXEL: urn:pixel:DataSource:itmnf:VesselCall)
Data Description	List of vessel calls for ships reaching or leaving ASPM or currently located at ASPM.
Data Origin	Data is extracted from the ASPM’s website (http://www.porto.monfalcone.gorizia.it/sailinglist.asp).
Data Format	HTML webpage
Integration	By means of a dedicated agent
Agent Name	Insiel Agent – VesselCall process https://gitpixel.satrdlab.upv.es/paolo.casoto/PIXEL_INSIEL/src/master/PIXELInsielAgent
Data model	VesselCall
Update frequency	Daily
Used by	Intermodal model, PAS

5.4.Functionalities provided, models and algorithms

The table below summarises the functionalities that have been made available to the port in the trial. The fact of having deployed those functionalities (based on that series of agents and not others) directly maps to the needs expressed in the use-case stories and the pilot objectives (see Context Review above). The table below also refers to the models that have been installed and run using the data fed by the agents described above.

Table 20. Functionalities deployed in APT – task T7.3

Functionality	Model	Inputs	Results shown
Energy consumption - temporal line	PAS model	Vessel calls, port parameters	A temporal line (evolution graph) with the energy consumed, separated per type of fuel (diesel, electricity...)
Expected trucks with respect to vessel calls	Intermodal transport model	SILI, SDAG_ACS, vessel calls	
Estimated Time of Departure prediction	Traffic Predictive Algorithm	Vessel calls, TrafficFlowObserved	Estimated time of departure of each vessel that is operated in the port. The visualisation is similar to the Gantt.
PEI	PEI model	eKPIs	(reported in section 8 below)

In the following sub-sections, there are some evidences and reflections over the results of the deployment of these functionalities. In the scope of WP7, the successful deployment of all the models in the table (= providing satisfactory, reasonable results) was a sufficient condition to indicate the T7.3 task as finalised. However, this “handover” of running models in the platform was complemented with an explanation of the usage of each model and the interpretation of the results. This “handover” and explanation took place on the Plenary Meeting held virtually at the beginning of June 2021.

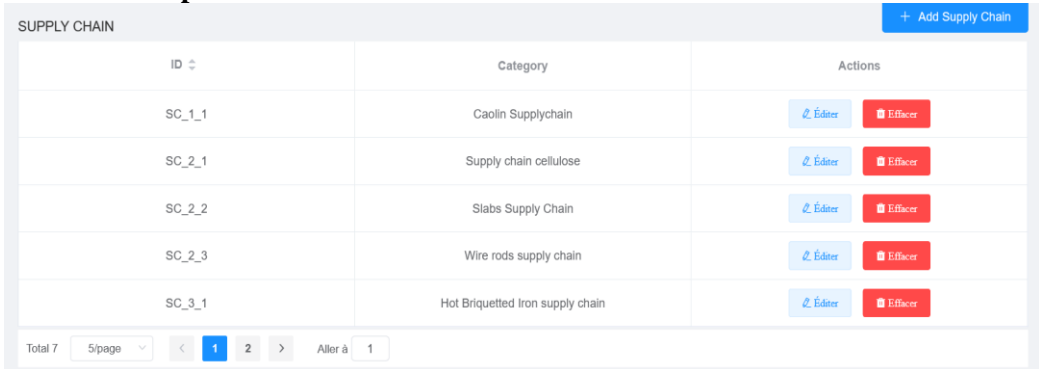
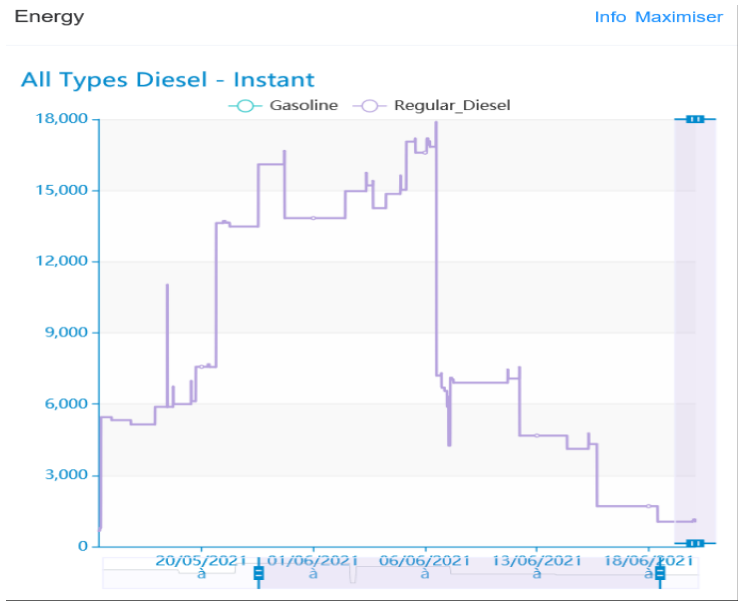
The way that those evidences are organised in the sub-sections below is **per model** (2nd column of table above). PEI is reported in Section 8 of the document.

5.4.1. PAS model

A detailed description of the PAS model, its rationale and structure are provided at paragraph 4.3. The PAS model has been proficiently applied to ASPM/APT in order to evaluate how port operations (related with vessel calls) impact in terms of both energy consumption and pollutant emissions. PAS has been applied on equipment and tools available at APT yards; a research activity has been performed in order to collect technical data about each different equipment involved in loading and unloading activities.

Table 21. Model “PAS Model” Port of Monfalcone

Model name	PAS model
Model Description	A detailed description of the model is provided at paragraph 4.3.
Inputs requested	Input data of the PAS model are mainly based on: i) vessel planning (arrival date, cargo type and tonnage), ii) activity data (details related to the transfer of cargo), iii) operational data (refer to the technical specifications of engines and equipment used) iv) emission source data (related to emissions factors of engines and equipment).

Outputs generated	The Port Activities Scenario (PAS), combined with the energy model and emission factors, allows to model port supply chains. The output of this model can be used as an input for evaluating the energy consumption (detailed information can be found at section 4.3.1)
Integration	Docker
Description of the model visualisations	<p>PAS Forms inputs</p>  <p><i>Figure 14. APT Pas model - PAS Form input</i></p> <p>Energy Output based on PAS model</p>  <p><i>Figure 15. APT Pas model - Energy output</i></p>

5.4.2. Intermodal transport model

Table 22. Model “Intermodal transport model” Port of Monfalcone

Model name	Intermodal transport model
Model Description	The model is aimed at providing information about expected trucks reaching ASPM with respect to planned vessel calls for a specific set of cargo types (which require direct transportation without storage of goods in warehouses) and forecasting congestions, which could happen inside the port area).

Inputs requested	SILI, SDAG_ACS and vessel calls.
Outputs generated	Custom data model representing current status of both APT and SDAG parking lots and forecasted availability.
Integration	Data are provided by means of agents
Description of the model visualisations	Visualizations are aimed at describing the expected number of trucks, the expected availability of parking lots (both ASPM and SDAG) and the possibility of congestion to take place according to the expected number of trucks reaching the port.

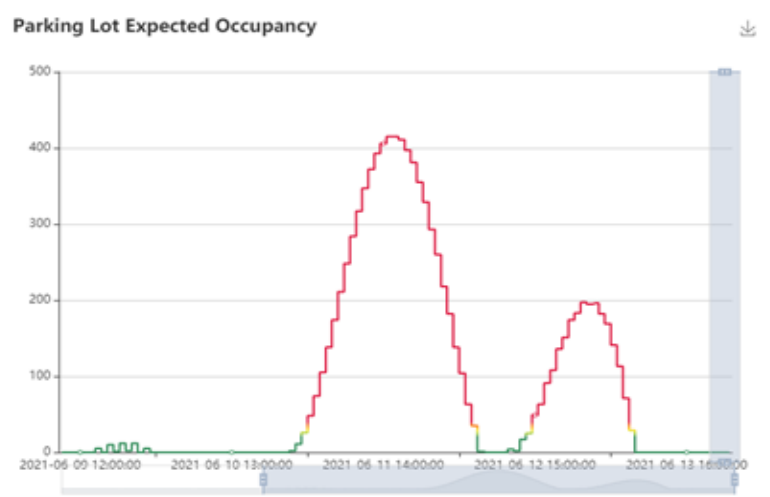


Figure 12. APT Intermodal transport model - parking lot expected occupancy output

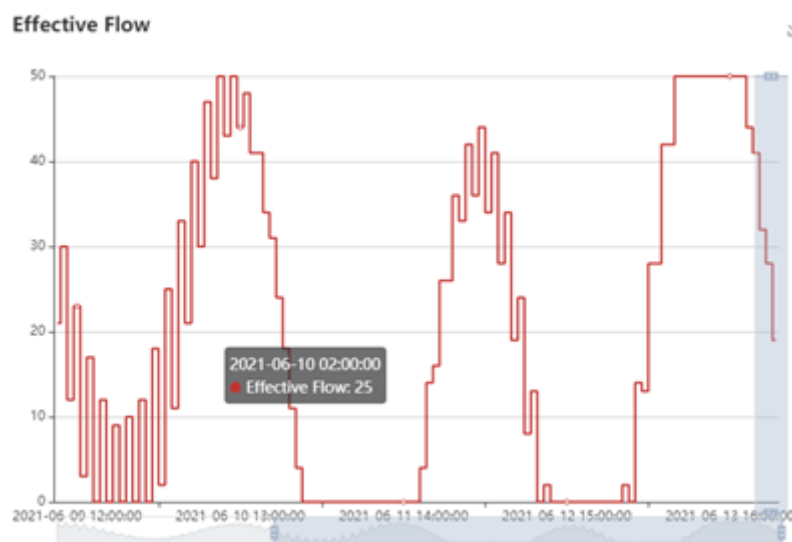


Figure 13. APT Intermodal transport model - effective flow output



Figure 14. APT Intermodal transport model max flow output

5.4.3. Traffic Predictive Algorithm

The SILI platform monitors continuously the traffic of trucks and car along the highways of Friuli Venezia Giulia (including border areas between Italy and Austria and between Italy and Slovenia) in order to:

- Evaluate the amount of traffic currently on the highway.
- Track how dangerous goods are moved along the highway.

In particular, with respect to PIXEL's goals, two spots observed by the SILI platform are particularly interesting: the gate used to access ASPM and the gate located near the SDAG facility.



Figure 15. Port gates in ASPM. Vehicle tracking

The task as part of the use case is aimed at making estimation about future traffic congestion at both port's gates and highway, considering recent and historical data. Besides the historic, other relevant data, potentially correlated, was also analysed: seasonality, weather, traffic status near the city.


The actions taken to accomplish the task were the following:

- a. **Import (near) real-time and historical data in the PIXEL platform (1)**, by using the already described SILI process, included inside the INSIEL Agent.
- b. **Deploy the traffic predictive algorithm (2)**. An algorithm was developed to make initial tests, which is based on *Prophet* libraries. Afterwards, the algorithm was coded and Dockerized following the PIXEL OT framework approach, so that it can be published and executed within the PIXEL platform on the whole set of measurement posts constituting the SILI platform.
- c. **Add specific visualizations for the Predictive Algorithm (3)** in order to provide to both ASPM and SDAG's operators with an effective visualization of forecasted data.

A detailed description of the Traffic Prediction Algorithm is included in D4.4; due to the modularity of the PIXEL's platform, such an algorithm can be proficiently moved across different ports in a seamless way.

Table 23. Algorithm "Prediction algorithm" Port of Monfalcone

Model name	Traffic prediction algorithm
Model Description	Allows estimating the traffic at each of the monitoring spots of the SILI platform located along the regional highway;

Inputs requested	Current and historical set of TrafficFlowObserved items at each monitoring spot. (recommended at least one year)
Outputs generated	TrafficFlowObserved Items with the predictions (by hour) for the next 5 days, for each monitoring spot. The result is stored into the specific index arh-lts-trafficflowobserved.
Integration	Trucks identified by each monitoring spot (aggregated on an hourly basis) are converted into TrafficFlowObserved items in the PIXEL platform by means of the Insiel Agent. Data is provided by the SILI platform with a delay of one day. The model has been fully integrated inside the ASPM infrastructure.
Description of the model visualisation	Three visualizations are provided, each one for a different analytic purpose
	<p>Map view of the traffic volume at port gates and in several defined points in the city</p>  <p><i>Figure 16. APT Traffic prediction algorithm - port gates and city traffic volume map</i></p>

Traffic monitor at port gate with predictions on a 7-day horizon

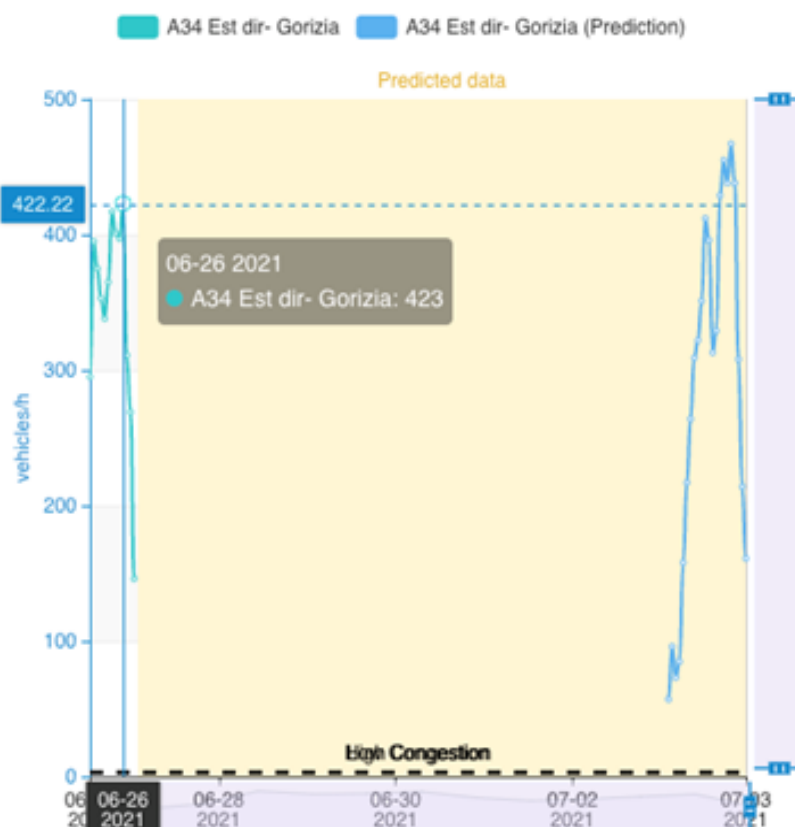


Figure 17. APT Traffic prediction algorithm - port gates 7 days traffic monitor

Table of traffic information with predictions

Gates ▾	From:	To:	Intensity
Prosecco Di rezione TS	03/07/2021 23:00	04/07/2021 00:00	612
Prosecco Di rezione TS	03/07/2021 22:00	03/07/2021 23:00	652
Prosecco Di rezione TS	03/07/2021 21:00	03/07/2021 22:00	756
Prosecco Di rezione TS	03/07/2021 20:00	03/07/2021 21:00	1031
Prosecco Di rezione TS	03/07/2021 19:00	03/07/2021 20:00	1315
Prosecco Di rezione TS	03/07/2021 18:00	03/07/2021 19:00	1412
Prosecco Di rezione TS	03/07/2021 17:00	03/07/2021 18:00	1345

Figure 18. APT Traffic prediction algorithm - traffic information with predictions

6. Port-city integration trial – Port of Thessaloniki

6.1. Context review

The use case of the Port of Thessaloniki focuses on the interoperability of city and port in freight traffic and on the pollution impact towards the city. The three main objectives of the pilot are:

1. To optimize the traffic between the city and the port area and thus alleviate congestion or bottlenecks, caused by its operations.
2. To measure air and noise pollution thus being able to establish mitigation actions to reduce its environmental impact while at the same time, strengthen its relationships with the city of Thessaloniki.
3. To integrate existing systems and devices (legacy and/or new) with the PIXEL platform.

As secondary objectives, the Port of Thessaloniki aims to gain insight about:

- Inbound and outbound traffic flows.
- Overall environmental footprint of the port.

6.2. Platform infrastructure

All the PIXEL installations in ports are recommended to have two servers:

- **Server 1 = VM 1** = named as **PUBLIC**. It will provide access from the outside if the port wants this feature. In this case, a public IP address or an equivalent mechanism (NAT access) will be required
- **Server 2 = VM 2** = named as **CORE**. IT will include most of the PIXEL core components according to its architecture.

THPA has installed the following:

- VM1 = 4 cores, 16 GB RAM, 300 GB HDD,
- VM2 = 4 cores, 16 GB RAM, 200 GB HDD

The servers/VMs selected have the following software installed:

- **Ubuntu server 18.04 LTS**: Ubuntu is the preferred Linux distribution where the PIXEL platform has been mostly tested.
- **OpenSSH server**: this will allow access for the server to proceed with the installation.

6.3. Data sources

In this section, all data sources integrated into the Pilot of Thessaloniki are described. The integration strategy used in any case depends on the data source.

6.3.1. Vessel Calls

Table 24. Data source “Vessel calls planned” Port of Thessaloniki

Data Name	Vessel Calls
Data Description	Thessaloniki Port vessel calls (past calls). This data comes from the API of THPA, in a JSON format. It contains both historic (since 2018) to future (forthcoming) data of the vessels operated in the port.
Data Origin	ThPA API

Figure 19. Vessels calls THPA data source

6.3.2. Gates Traffic Data

Table 25. Data source “Gates Traffic data” Port of Thessaloniki

- When someone enters port, and from which Gate.
- When someone leaves the port, and from which Gate.
- Dwell time spent within the po

	<pre> { "id": "2245120", "vehicle_type": "Car", "brand": "SEAT ", "model": "ARONA ", "colour": "WHITE", "entry_time": "-", "exit_time": "5\\7\\2021 06:01:53", "entry_gate": null, "exit_gate": "Gate 16 - Exit #2", "dwell_time": "", "notes": "" }, </pre> <p><i>Figure 20. Traffic at the gates data THPA</i></p>
Integration	NGSI Agent
Agent Name	thpa-ngsi-gatestraffic
Data model	Traffic Flow Observed (Fiware) https://github.com/smart-data-models/dataModel.Transportation/tree/master/TrafficFlowObserved
Update frequency	Hourly
Used by	Traffic Predictive Algorithm

6.3.3. Wind Data

Table 26. Data source “Wind data” Port of Thessaloniki

Data Name	Wind data
Data Description	Information gathered from the wind sensor installed in THPA. The data are provided to PIXEL via an exposed API that sends (every 2 seconds) values of angle and speed of the wind.
Data Origin	ThPA API
Data Format	JSON <pre> { "id": 109282513, "speed": 3.5, "angle": 307, "when": "5\\7\\2021 18:42:58" }, </pre> <p><i>Figure 21. Wind data source THPA</i></p>

Integration	NGSI Agent
Agent Name	thpa-ngsi-wind
Data model	Weather Observed (Fiware)
Update frequency	Every 2 seconds.
Used by	Air Pollution Model

6.4. Functionalities provided, models and algorithms

The table below summarises the functionalities that have been made available to the port in the trial. The fact of having deployed those functionalities (based on that series of agents and not others) directly maps to the needs expressed in the use-case stories and the pilot objectives (see Context Review above). The table below also refers to the models that have been installed and run using the data fed by the agents described above.

Table 27. Functionalities deployed in THPA – task T7.4

Functionality	Model	Inputs	Results shown
Traffic at the gates of the port (estimation in the short-term)	Traffic Predictive Algorithm	Traffic at the gates	Estimated time of departure of each vessel that is operated in the port. The visualisation is similar to the Gantt.
Noise pollution dispersion in a heatmap	Noise pollution	Port parameters (areas, ships, noise sources), worst+medium+best case of traffic.	Noise pollution dispersion in a heatmap
Noise pollution dispersion in a heatmap	Air pollution (AERMOD)	Air emission eKPIs, see Appendix E .	Noise pollution dispersion in a heatmap
PEI	PEI model	eKPIs	(reported in section 8 below)

In the following sub-sections, there are some evidences and reflections over the results of the deployment of these functionalities. In the scope of WP7, the successful deployment of all the models in the table (= providing satisfactory, reasonable results) was a sufficient condition to indicate the T7.2 task as finalised. However, this “handover” of running models in the platform was complemented with an explanation of the usage of each model and the interpretation of the results. This “handover” and explanation took place on the Plenary Meeting held virtually at the beginning of June 2021.

The way that those evidences are organised in the sub-sections below is **per model** (2nd column of table above). PEI is reported in Section 8 of the document.

6.4.1. Traffic Predictive Algorithm

The situation in THPA is that there is a variable congestion at the port gates, which is rather typical at ports. Currently the port already includes an RF identification service to track vehicles going inside or outside the port, and it is storing such historical data. However, no analytics are performed and no exploit any potential knowledge behind such information.



Figure 22. Port gates in THPA. Vehicle tracking

The task as part of the use case implied making estimation about future traffic congestion at both gates, considering recent and historical data. Besides the historic, other relevant data, potentially correlated, was also analysed: seasonality, weather, traffic status near the city and vessel calls.

Furthermore, how the data is treated should also follow a homogeneous approach, allowing the use of FIWARE data models for inputs and outputs, ending up in a way of facilitating the visualization of results.



Figure 23. THPA traffic congestion illustration

The actions taken to accomplish the task were the following:

- d. **Import (near) real-time and historical data in the PIXEL platform (1).** The developed NGSI agent follows a common data format based on the FIWARE data model (*TrafficFlowObserved*) and is scheduled to execute every hour. The IH persists such data.
- e. **Deploy the traffic predictive algorithm (2).** An algorithm was developed to make initial tests. The algorithm is based on Facebook Prophet libraries. Afterwards, the algorithm was coded and Dockerized following the PIXEL OT framework approach, so that it can be published and executed within the PIXEL platform.
- f. **Add specific visualizations for the Predictive Algorithm (3).** Every model or algorithm produces data in a specific way that requires specific ways of visualizing such data. This was done also for this algorithm.

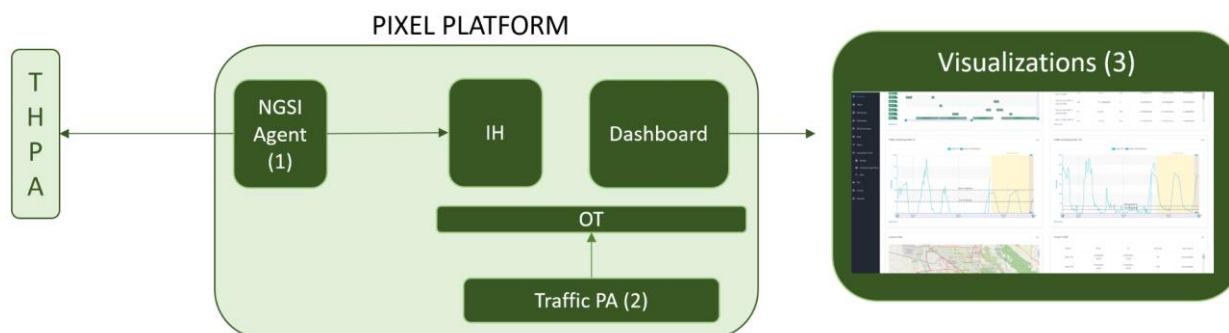


Figure 24. THPA prediction algorithm. Actions taken

As a result of the previous actions, the PIXEL platform is updated every hour with new estimation for the upcoming 5 days. The gate managers can visualize in various ways the available results, establishing different thresholds for each gate.

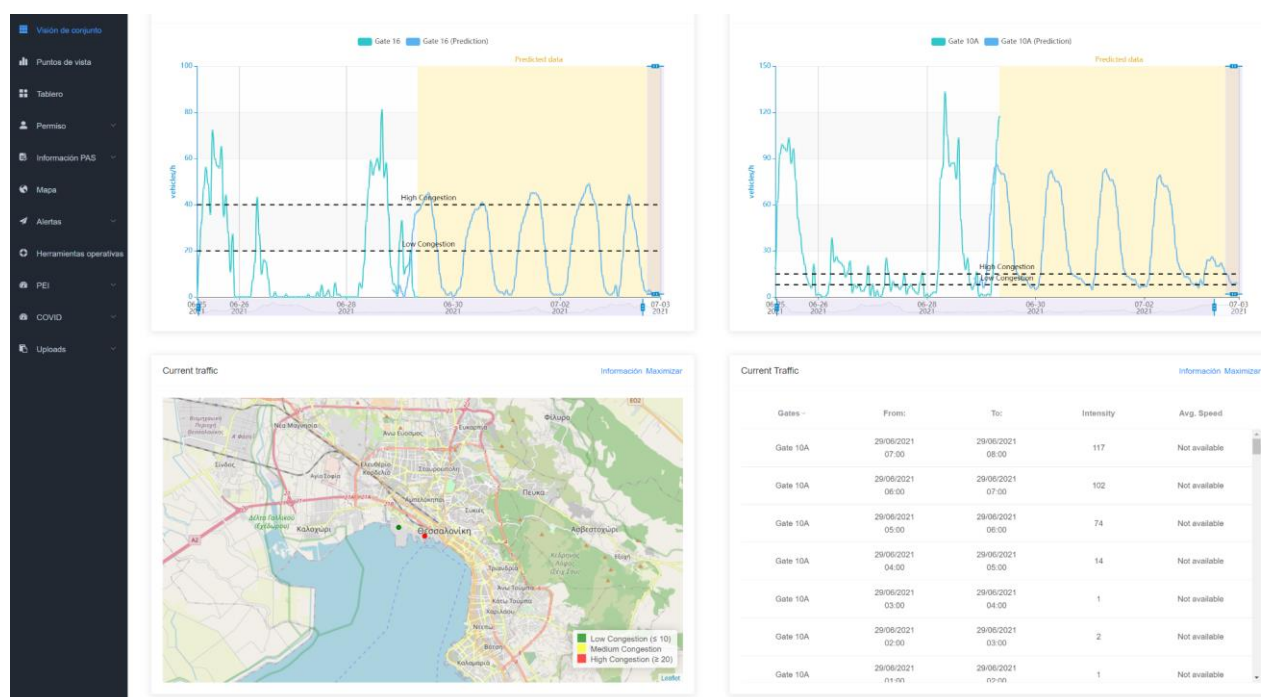
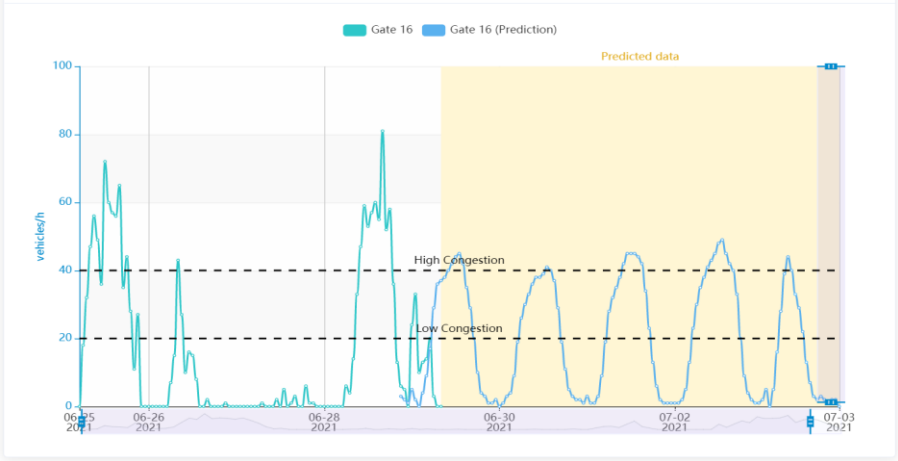
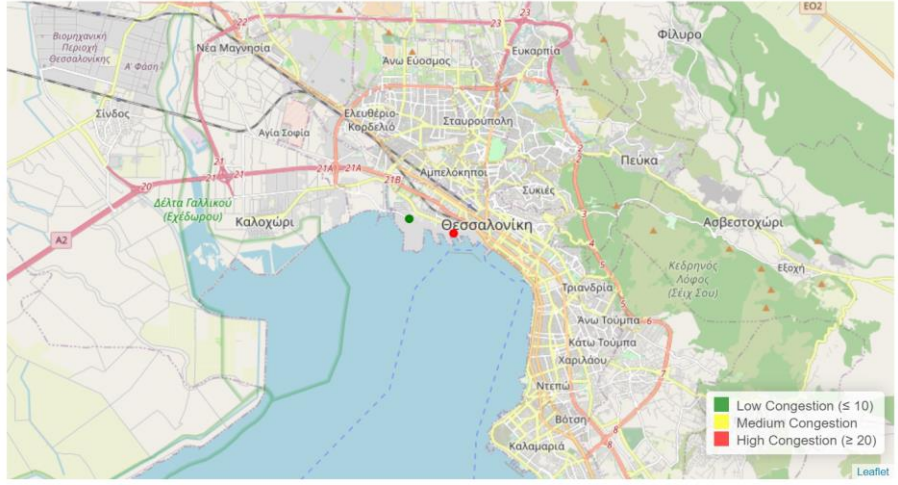
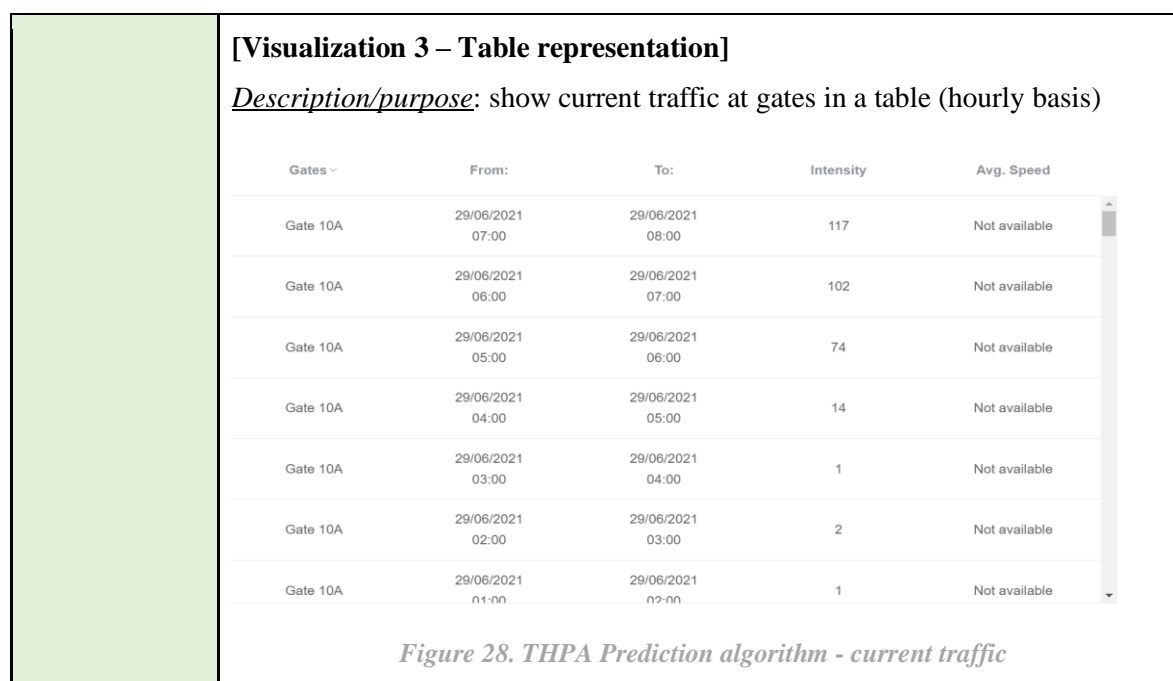


Figure 25. THPA traffic prediction. Results visualizations in the Dashboard

Table 28. Algorithm “Prediction algorithm” Port of Thessaloniki

Model name	Prediction algorithm
Model Description	Allows estimating the traffic at both gates in THPA (Gate 16 and Gate 10A)
Inputs requested	Current and historical set of <i>TrafficFlowObserved</i> items at both gates (recommended at least one year)
Outputs generated	<i>TrafficFlowObserved</i> Items with the predictions (by hour) for the next 5 days, for each gate. The result is stored in a specific index

Integration	Vehicles at both gates are given by THPA API and are converted into <i>TrafficFlowObserved</i> items in the PIXEL platform by means of an NGSI Agent)
Description of the model visualisations	<p>[Visualization 1 – Graph representation]</p> <p><i>Description/purpose:</i> show recent data and future estimation/prediction. Thresholds can be set per gate while creating the visualization</p>  <p><i>Figure 26. THPA Prediction algorithm - port gates traffic prediction</i></p>
	<p>[Visualization 2 – Map representation]</p> <p><i>Description/purpose:</i> show congestion status (low/medium/high) on geolocated traffic gates</p>  <p><i>Figure 27. THPA Prediction algorithm - congestion status map</i></p>



6.4.2. Noise Model


From the work carried out in WP4, Predictor-LimA was chosen as the best way to perform noise simulations. However, as a proprietary software, it was not possible to be integrated in the PIXEL platform and be run as a PIXEL model.

The alternative approach was to extend the functionality of the PIXEL platform to import the results of Predictor-LimA. This software is able to output KML files, showing something similar to a heatmap of noise distribution according to some configurations specific for ports.

Several best, medium and worst cases were executed with Predictor-LimA and imported in the PIXEL platform as KML files. A visualization of such files was also created in the Dashboard.

Table 29. Model “Noise model (import facility)” Port of Thessaloniki

Model name	Noise model (import facility)
Model Description	Allows importing KML results from the execution of Predictor-LimA
Inputs requested	None (import facility)
Outputs generated	None
Integration	The KML file from Predictor-LimA is imported in the PIXEL platform through a (web) form; then it can be visualized in the Dashboard with Leaflet enabled support.

Description of the model visualisations	[Visualization 1 – KML representation] <i>Description/purpose:</i> show KML result from the execution of Predictor-LimA. The KML is georeferenced and shows the different noise sources employed.
Why has this model not been integrated?	<p>The model has been partially integrated (result import).</p>  <p><i>Figure 29. THPA Noise model - Noise level map layer</i></p>

6.4.3. Air pollution model

From the work carried out in WP4, AERMOD was chosen as the best way to perform air pollution estimations. However, AERMOD is not a single program, but a set of programs, including AERMAP, AERMET and AERPLOT among others.

The integration of such a model within the PIXEL platform was explored and even reached up to a certain limit: cross-compilation was successful in Linux and a Docker was created containing most of the functionalities; however, some private libraries employed by AERPLOT were not possible to cross-compile for Linux and no support was provided by the code owners.

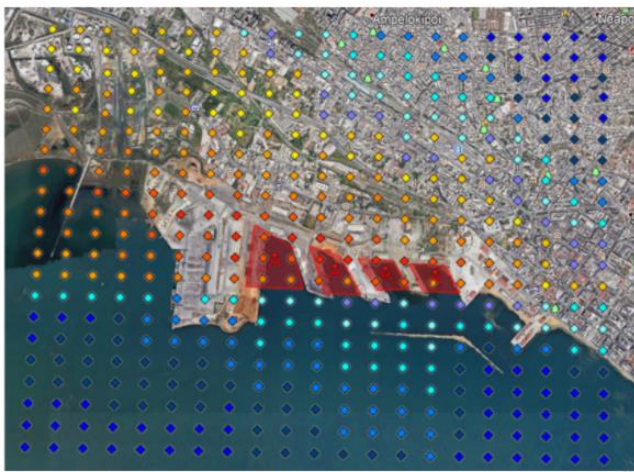
The alternative approach was to extend the functionality of the PIXEL platform to import the results of AERMOD. This software is able to output KMZ files, showing something similar to a grid of pollution concentration for a given air source.

For the details on the execution of AERMOD for THPA in PIXEL, see **Appendix E**.

Several worst cases were executed with AERMOD and imported in the PIXEL platform as PNG files. A visualization of such files was also created in the Dashboard.

Table 30. Model “Air pollution model (import facility)” Port of Thessaloniki

Model name	Air pollution model (import facility)
Model Description	Allows importing PNG results from the execution of AERMOD

Inputs requested	None (import facility)
Outputs generated	None
Integration	The KMZ file from AERMOD (AERPLOT) is converted into a PNG file together with the table outputs from AERMOD; then it is imported in the PIXEL platform through a (web) form, that allows metadata insertion; then it can be visualized in the Dashboard.
Description of the model visualisations	<p>[Visualization 1 – PNG representation]</p> <p><i>Description/purpose:</i> show PNG result from the execution of AERMOD. The PNG includes a table highlighting the maximum value to be checked for maximum legal allowed values. It includes a link to a web page with these values.</p>
Why has this model not been integrated?	<p>The model has been partially integrated (result imported).</p>  <p>*** MODELOPTS: Reg@FAULT CONC ELEV RURAL</p> <p><i>Figure 30. THPA Air pollution model - Air pollution level map layer</i></p>

7. Port-city integration trial – Port of Piraeus

7.1.Context review

The passenger port at PPA is very near to the city of Athens, and there are residential areas very near the port. The large number of cruise ships that the port normally receives during the holiday periods, affect the living of the people living around. When lots of cruises arrive and it coincides with current traffic in the city, the touristic buses are not able to reach the Acropolis in a reasonable time for tourists to visit it. It also implies more traffic congestion, noise pollution and overall environmental impact. This use case does not involve only cruise but also container and coastal (passenger) traffic operations.

The models used in this trial are oriented to monitor the impact of the port on the city and try to anticipate/predict serious environmental impacts, as well as traffic problems for both citizens and tourists.

7.2. Platform infrastructure

For this trial two virtual machines running in MS Azure were deployed.

Virtual Machine (B4MS) specs are:

- 4 vCPU(s),
- 16 GB RAM);
- OS: Linux – Ubuntu;
- Managed Disks, Standard HDD, S20 Disk Type 2 Disks

Moreover, an FTP server has been added to the infrastructure in order to upload and provide data coming from third party systems and sensors. The information stored periodically in the FTP is consumed by different agents to store the data into the PIXEL platform in the proper format.

7.3. Data sources

In this section, all data sources integrated into the Port of Piraeus PIXEL infrastructure are described. The integration strategy used in any case depends on the data source.

7.3.1. Vessel Calls

Table 31. Data source “Vessel calls planned” Port of Piraeus

Data Name	Vessel Calls
Data Description	This data source is based on AIS Data, enhanced by Marine Traffic.
Data Origin	MarineTraffic API
Data Format	JSON
Integration	NGSI Agent
Agent Name	ppa-ngsi-vesselcall
Data model	VesselCall
Update frequency	daily
Used by	PAS Model

7.3.2. Dark Sky

Table 44. Data source “Dark Sky” Port of Piraeus

Data Name	Dark Sky
Data Description	Weather information

Data Origin	Information retrieved from the Darksky API
Data Format	Rest API - JSON format
Integration	Using Agents
Agent Name	WeatherAgent
Data model	Data Model used: WeatherObserved Repository: https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/Weather/WeatherObserved/schema.json
Update frequency	Agent is executed each 15 minutes
Used by	This information is used by the Traffic model in order to have weather information into account for calculating the future traffic congestion

7.3.3. Cruise Ships

Table 32. Data source “Cruise ships” Port of Piraeus

Data Name	Cruise ships
Data Description	Ships arriving to the port on a monthly basis
Data Origin	spreadsheet uploaded monthly to the FTP
Data Format	.xlsx
Integration	using Agents
Agent Name	CruiseAgent
Data model	Data Model used: VesselCalls Repository: https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/VesselCall/schema.json
Update frequency	monthly
Used by	This information is used by the Traffic model in order to have ship information and number of busses needed for calculating the future traffic congestion

7.3.4. Here maps

Table 33. Data source “Here maps” Port of Piraeus

Data Name	Here maps
Data Description	Current congestion in the roads near PPA
Data Origin	Online API
Data Format	JSON
Integration	Using Agents
Agent Name	HereAgent
Data model	Data Model used: TrafficFlow Repository: https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/Traffic/TrafficFlow/schema.json
Update frequency	15 minutes
Used by	This information is used by the Traffic model in order to have ship information and number of busses needed for calculating the future traffic congestion

7.3.5. Meteo

Table 34. Data source “Meteo” Port of Piraeus

Data Name	Meteo
Data Description	Hourly information about the weather and pollutants at port. This information comes from a real weather and air quality station that provides data in real time.
Data Origin	Files uploaded to an FTP server.
Data Format	XSLX
Integration	Using Agents
Agent Name	MeteoAgent
Data model	Data model used: AirQualityObserved Repository:

	https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/AirQualityObserved/schema.json
Update frequency	Hourly
Used by	Not used by any model. The added value here is that the information about the weather and pollutants can be shown as raw data in the platform using custom visualizations and the map view.

7.4. Functionalities provided, models and algorithms

The table below summarises the functionalities that have been made available to the port in the trial. The fact of having deployed those functionalities (based on that series of agents and not others) directly maps to the needs expressed in the use-case stories and the pilot objectives (see Context Review above). The table below also refers to the models that have been installed and run using the data fed by the agents described above.

Table 35. Functionalities deployed in PPA – task T7.4

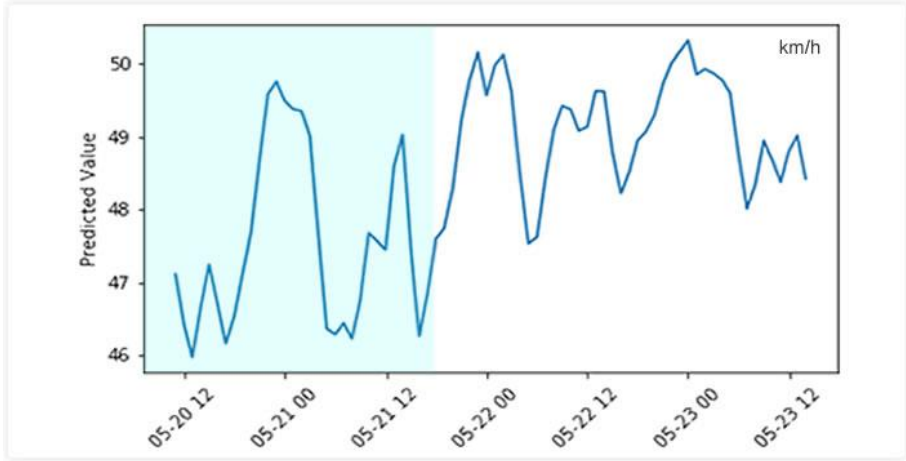
Functionality	Model	Inputs	Results shown
Traffic at the gates of the port (estimation in the short-term)	Traffic Predictive Algorithm	DarkSky, HereMap, Cruise ships	Estimated time of departure of each vessel that is operated in the port. The visualisation is similar to the Gantt.
Noise pollution dispersion in a heatmap	Noise pollution	Port parameters (areas, ships, noise sources), worst+medium+best case of traffic.	Noise pollution dispersion in a heatmap
Noise pollution dispersion in a heatmap	Air pollution (AERMOD)	Air emission eKPIs, see Appendix E .	Noise pollution dispersion in a heatmap
PEI	PEI model	eKPIs	(reported in section 8 below)

In the following sub-sections, there are some evidences and reflections over the results of the deployment of these functionalities. In the scope of WP7, the successful deployment of all the models in the table (= providing satisfactory, reasonable results) was a sufficient condition to indicate the T7.2 task as finalised. However, this “handover” of running models in the platform was complemented with an explanation of the usage of each model and the interpretation of the results. This “handover” and explanation took place on the Plenary Meeting held virtually at the beginning of June 2021.

The way that those evidences are organised in the sub-sections below is **per model** (2nd column of table above). PEI is reported in Section 8 of the document.

7.4.1. Traffic Predictive Algorithm

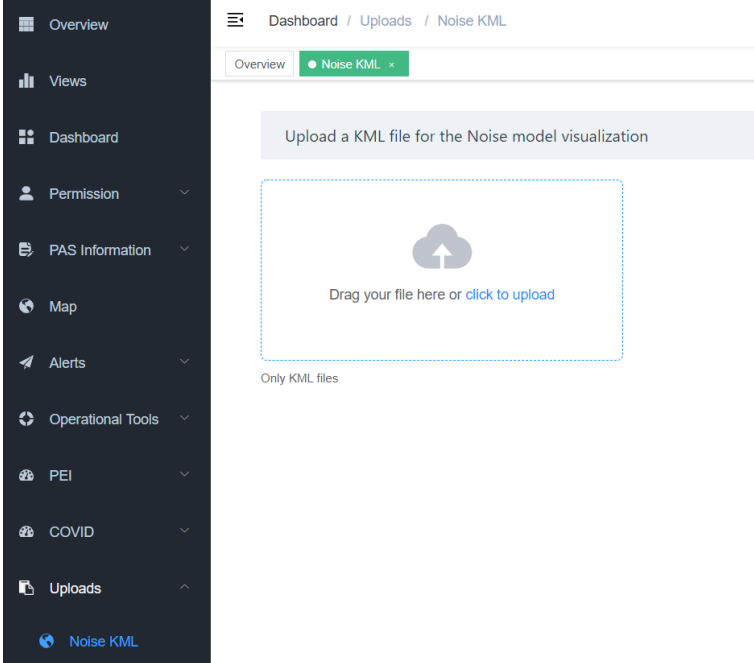
Table 36. “Traffic” Predictive Algorithm Port of Piraeus

Model name	Traffic Congestion
Model Description	This model predicts the Traffic congestion at some points near the port, based on the number of cruise ships arrivals planned
Inputs requested	Cruise Ships & busses needed, Real time traffic information (Here) and weather information (dark sky)
Outputs generated	The prediction of traffic congestion in the road adjacent to the port in the future days in an hourly frequency.
Integration	<p>The data requested by the model is provided by three agents integrated in the platform:</p> <ul style="list-style-type: none"> - Dark Sky - Cruise ships - Here maps
Description of the model visualisations	<p>Linear graph with congestion predictions</p> <p>This graph represents congestion (Velocity of the vehicles) in the road adjacent to the port over time, from a few days - hours before the current moment, to future predictions (days - hours) considering the expected port activity and historical data.</p> <p>Thanks to this visualization the port is aware not only of the past congestion situation outside the port, but also of the expected congestion. This information is very useful for the port because it can better organize the arrival and departure of the buses used to transport tourists into the city</p>  <p>Figure 31: PPA Velocity prediction</p>

7.4.2. Noise Model

Table 50. Model “Noise model” Port of Piraeus

Model name	Noise model
Model Description	<p>The model allows the port to simulate the noise dispersion. With this information, the port can understand the effect that port activity has on emissions and its environmental impact.</p> <p>As described in the WP4, a commercial software called “Predictor-LimA” was chosen as the best way to perform noise simulation. Due to its licence, it is not possible to fully integrate the model</p> <p>The model cannot be integrated into the platform and it is not possible to use real-time data. In order to run the model, complex data preparation is required. For the pilot, the worst and the best scenarios have been calculated. These scenarios consider unfavourable and favourable situations based on weather conditions and port activities. A visualization of such files was also created in the Dashboard.</p>
Inputs requested	<p>The model needs the following inputs:</p> <ul style="list-style-type: none"> ● Weather information, ● Geographical information ● Port activity and ships (noise emission)
Outputs generated	The software Predictor-lima generates simulations of the noise dispersion that can be visualized in the program. But, in order to show this information integrated in the PIXEL platform, these simulations should be exported in several formats.
Integration	<p>This model cannot be integrated. This model is executed outside the PIXEL platform and then the outputs/results are imported.</p> <p>In this case the simulation in KML format is imported in the PIXEL platform through a (web) form; then it can be visualized in the Dashboard in a map visualization.</p>

	 <p style="text-align: center;"><i>Figure 32. ATP Noise KML importation form</i></p>
Description of the model visualisations	<p>[Visualization 1 – KML representation]</p> <p><u>Description/purpose:</u> show KML result from the execution of Predictor-LimA. The KML is georeferenced and shows the different noise sources employed.</p> <p>Thanks to the model the port has a better understanding of the noise produced, as well as the source, providing valuable information that allows the port to understand the effects of its activity. In the future, if the port decides to implement any mitigating activity, this can be monitored. For example, to reduce noise, the port can electrify several docks, so Cruise ships can turn off their engines. If we re-run the model with the new data, we can compare the noise model before and after the action.</p>

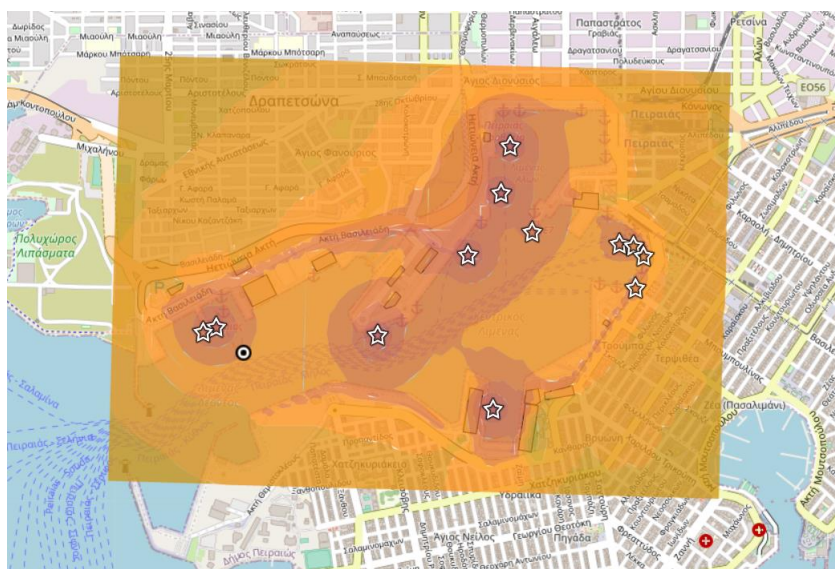


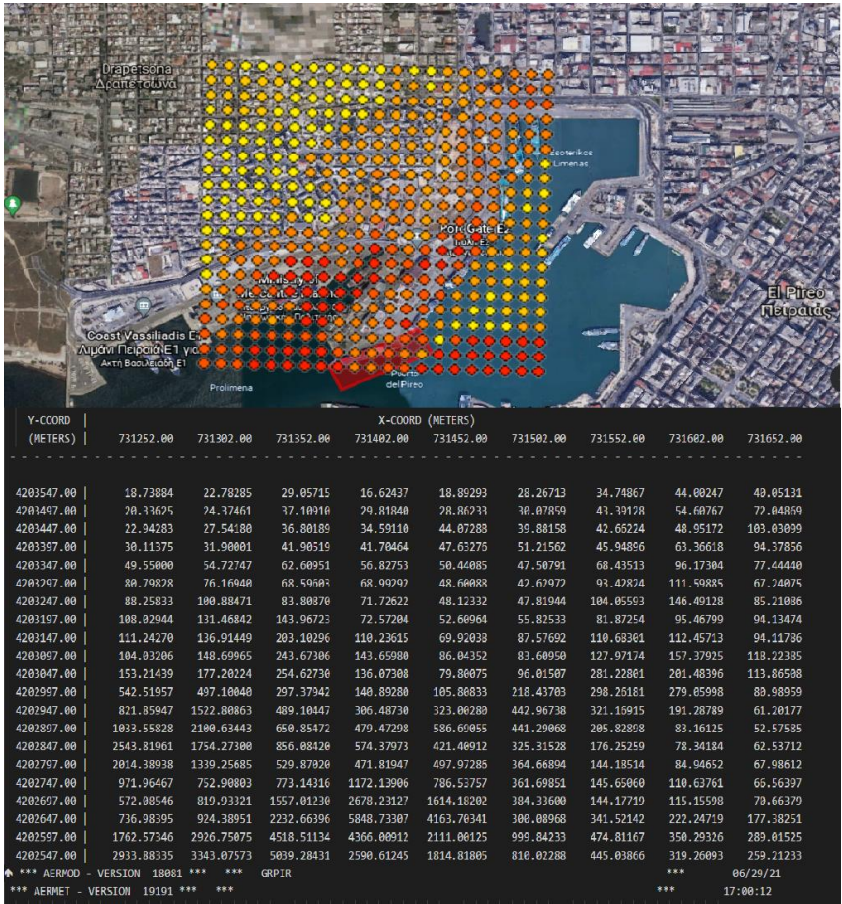
Figure 33. ATP Noise model - KML representation

Why has this model not been integrated?	Due to the licence of the Predictor limA, it is not possible to fully integrate the model.
--	--

7.4.3. AERMOD

Table 37. Model “Air Pollution model” Port of Monfalcone

Model name	Air pollution model
Model Description	<p>It is an air dispersion model based on AERMOD. This model is not a single program, but a set of programs, including AERMAP, AERMET and AERPLOT among others.</p> <p>Several worst cases were executed with AERMOD and imported in the PIXEL platform as PNG files. A visualization of such files was also created in the Dashboard.</p>
Inputs requested	<ul style="list-style-type: none"> • Weather information • Terrain information • Pollutant emission sources related with the port activity
Outputs generated	The simulation of the Air pollutant dispersion based on the inputs provided.
Integration	<p>The integration of such a model within the PIXEL platform was explored and even reached up to a certain limit: cross-compilation was successful in Linux and a Docker was created containing most of the functionalities; however, some private libraries employed by AERPLOT were not possible to cross-compile for Linux and no support was provided by the code owners.</p> <p>Finally, the KMZ file from AERMOD (AERPLOT) is converted into a PNG file together with the table outputs from AERMOD; then it is imported in the PIXEL</p>

	platform through a (web) form, that allows metadata insertion; then it can be visualized in the Dashboard.
Description of the model visualisations	<p>[Visualization 1 – PNG representation]</p> <p><i>Description/purpose:</i> show PNG result from the execution of AERMOD. The PNG includes a table highlighting the maximum value to be checked for maximum legal allowed values.</p>  <p>Figure 34. APT Air pollution model - PNG output of AERMOD execution</p>
Why has this model not been integrated?	A lot of work has been done in order to fully integrate this model into the PIXEL platform, but unfortunately it was impossible due to some compilation problems with some internal libraries used by AERPLOT. In the end, the model is executed outside the platform, and outputs of the model are imported and visualised in the PIXEL platform.

8. Transversal trial: Port Environmental Index

This section describes the deployment of the PEI at the four ports which are involved in PIXEL. Based on the results obtained, the ports could be compared in terms of environmental and sustainability performance. Data gathering has been the major challenge to implement the PEI in the four ports, and corrective actions and procedures for an effective PEI have been taken.

8.1. PEI development and integration methodology

The PEI model requires a specific analysis of the data available at each port in order to know how it could and should be used (merged, aggregated., etc.) in order to get the environmental KPIs (eKPIs). Note that data might come from heterogeneous sources at different frequency updates; sometimes data might even be typed manually.

The basic schema from an integration perspective is depicted in the Figure below

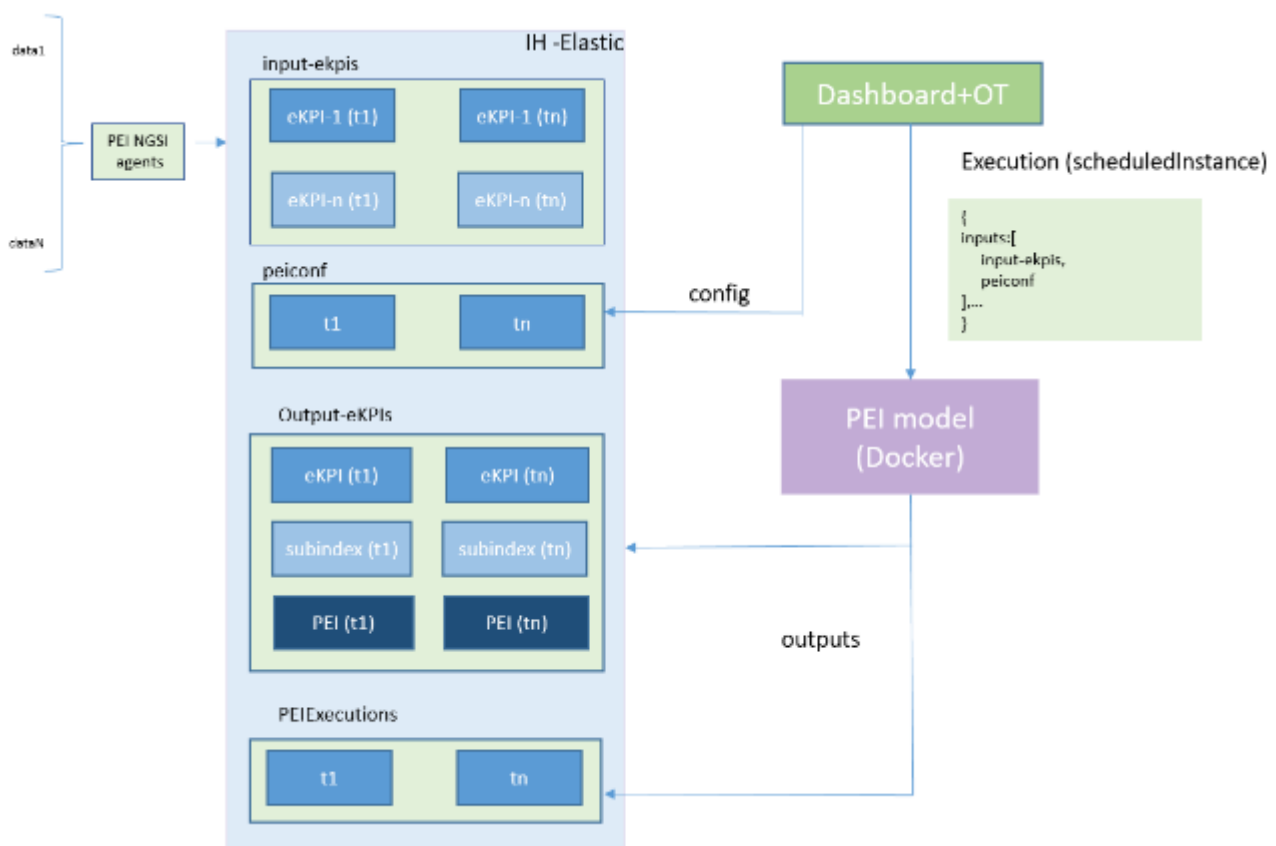


Figure 35. PEI overview

The process is as follows:

- A set of data is needed to calculate the PEI. However, every port will give a different list of input data from different sources and at a variable rate
- Depending on the source, data formats and frequency rate, a specific NGSI agent needs to be developed dealing with such requirements. The number of needed agents is unpredictable in advance, it depends on and varies significantly from port to port.
- Every Agent adapts the input data from the port in certain input eKPIs, following the FIWARE data model. Depending on the data, an agent may insert data every hour (e.g., light, noise) or every month (e.g., electricity usage). In other terms, the Information Hub (IH) will be storing input eKPIs with different granularities (from hour to month)

- Besides the input eKPIs serving as general input for the PEI model, there is additional data that refers to how the model should operate (normalization and weighting methods, etc.). Such information is also stored in the IH and can be edited by the port operator from the PIXEL Dashboard.
- From the Dashboard, once the data is available and the PEI model has been published, the port operator can execute the model, or even schedule it to be run every month (worst case granularity).
- The PEI model is executed and the results are stored also in the IH, this time under a different index (output-eKPIs in the Figure). Such eKPIs include the final eKPIs, which are calculated first, then the aggregated eKPIs (SEI, TEI, GEI) and finally the global eKPI (PEI). Once again, the format follows the FIWARE KPI data model.
- Besides the output eKPIs, additional information is also generated: Reliability Ratings (RR) and report documents.

Further details about inputs and outputs during the execution of the PEI model are presented in the sections below. As the general process is quite similar for all ports as long as the input EKPIs are properly provided, the model for one port (THPA) is described in more detail and will provide additional comments for the other ones.

8.2. Story and operative details

The development of the PEI as a composite indicator for the measurement of the negative impact that ports have on the environment was done as part of the WP5 and was described in all of the deliverables published as part of that work package (Deliverables 5.1, 5.2, 5.3 and 5.4). The task of choosing the relevant sub-indicators (here referred to as “environmental key performance indicators (eKPIs)”) had to be done having in mind two criteria:

1. The error in the evaluation of the ports’ impact on the environment had to be as low as possible
2. The chosen eKPIs had to be both objectively measurable and it had to be possible for the ports’ to obtain the required information in a way that is not too inconvenient (either technically, financially or both)

The process of the identification of those indicators was done by performing several steps. The first step included the pinpointing of the port activities that have impact on the environment and correctly identifying and relating the most important environmental aspects connected with those activities. Those important environmental aspects were named “significant environmental aspects (SEA)”, based on the terminology used in the applicable scientific sources, such as Darbra et al. 2005.

Key issue of the whole procedure was to determine which of the numerous environmental aspects commonly found in the literature and/or suggested by the experts are important enough to be labelled as “significant” and which are not. In order to consider different perspective on the possible solution of the issue, two seemingly opposite approaches were used:

1. “Scientific” approach, which consisted of reading the scientific papers and research reports related to the issue and picking out those that seemed to be most reliable and applicable for the purpose of the PEI development. Also, the frequency of various environmental aspects being described as “important” or “significant” in those papers had a major (although not necessarily conclusive) influence on the final decision.
2. “Practical” approach, which consisted of creating the questionnaire for the assessment of the importance of the individual environmental aspects and forwarding it to the four pilot ports (Ports of Bordeaux, Monfalcone, Piraeus and Thessaloniki). Their answers were carefully examined and, together with the information found in the literature, formed the basis for the final decision on the “significance” of individual environmental aspects and their inclusion in the PEI.

After both approaches were finished, it was concluded to use the following six environmental aspects to calculate the PEI:

- Emissions to the atmosphere (air emissions)
- Wastewater emissions
- Waste production

- Noise pollution
- Light pollution
- Odour pollution

It should be noted that the last environmental aspect (odour pollution) is a complicated one for the assessment of the significance, as it can be extremely important in some ports (such as fishing ports and the ports in which there is a lot of livestock/cattle transport) and relatively unimportant in others. Having that in mind, it was decided to mark the first five environmental aspects as “obligatory” for the assessment on each port and the odour pollution to be measured only in those ports that are affected by it. None of the pilot ports belonged to the latter category.

The next step in the procedure was to determine the most representative and relevant indicators (eKPIs) for each of the chosen significant aspects. The criteria for their choice were described in the deliverables created as part of the WP5 and are listed here again for convenience:

- Significance – same as the aspects related to the activities that can be both significant and insignificant, certain indicators can be related to the aspects, but they are not important enough to be included in the PEI. The similar approach was used as the one used for the determination of the significant of the aspects.
- Measurability – the chosen indicators should be measured either using real-time IoT equipment or by examining the data that is already regularly obtained by the ports.
- Representativeness – the environmental impacts of port activities should be clearly separated from the impacts not resulting from those activities (such as traffic and industries not related to the ports)
- Correlation

8.3. Definition of the model (configuration)

As commented before, the PEI requires a certain set of inputs that will be commented with more detail here. All these inputs (the access to them, in fact) are specified from a general perspective following the OT framework so as to enable the publication in the PIXEL platform.

PEI Tree Configuration

This is related to the way the PEI is created as a reverse tree, from the leaf nodes (EKPIs) to the ultimate branch (PEI), assigning various weights and performing different operations. Different levels and associated nodes are created in the form of a JSON file that is stored in the IH under a certain index.

From the point of view of the user, the Dashboard allows to configure and set up following values, as depicted in the Figure below.

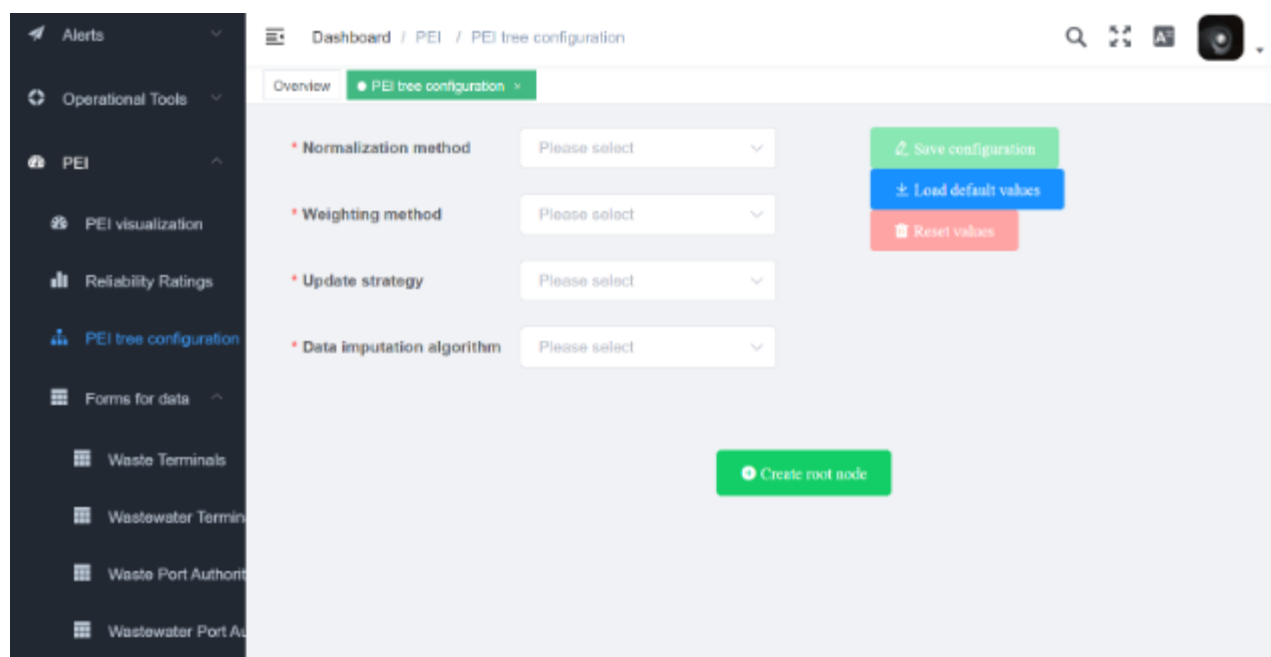


Figure 36. PEI Tree configuration from the Dashboard

The supported values are:

- **Normalization method:** standardisation (z-scores), re-scaling, distance to a reference port
- **Weighting method:** hand typed, equal weighting, budget allocation
- **Update strategy:** ask every time, replicate last value
- **Data imputation algorithm:** hot deck, cold deck, unconditional mean/median/mode

The user is able to load default values by pressing the blue button in the Dashboard. An automatic Tree is generated and displayed to the user. This configuration can be saved or reset.

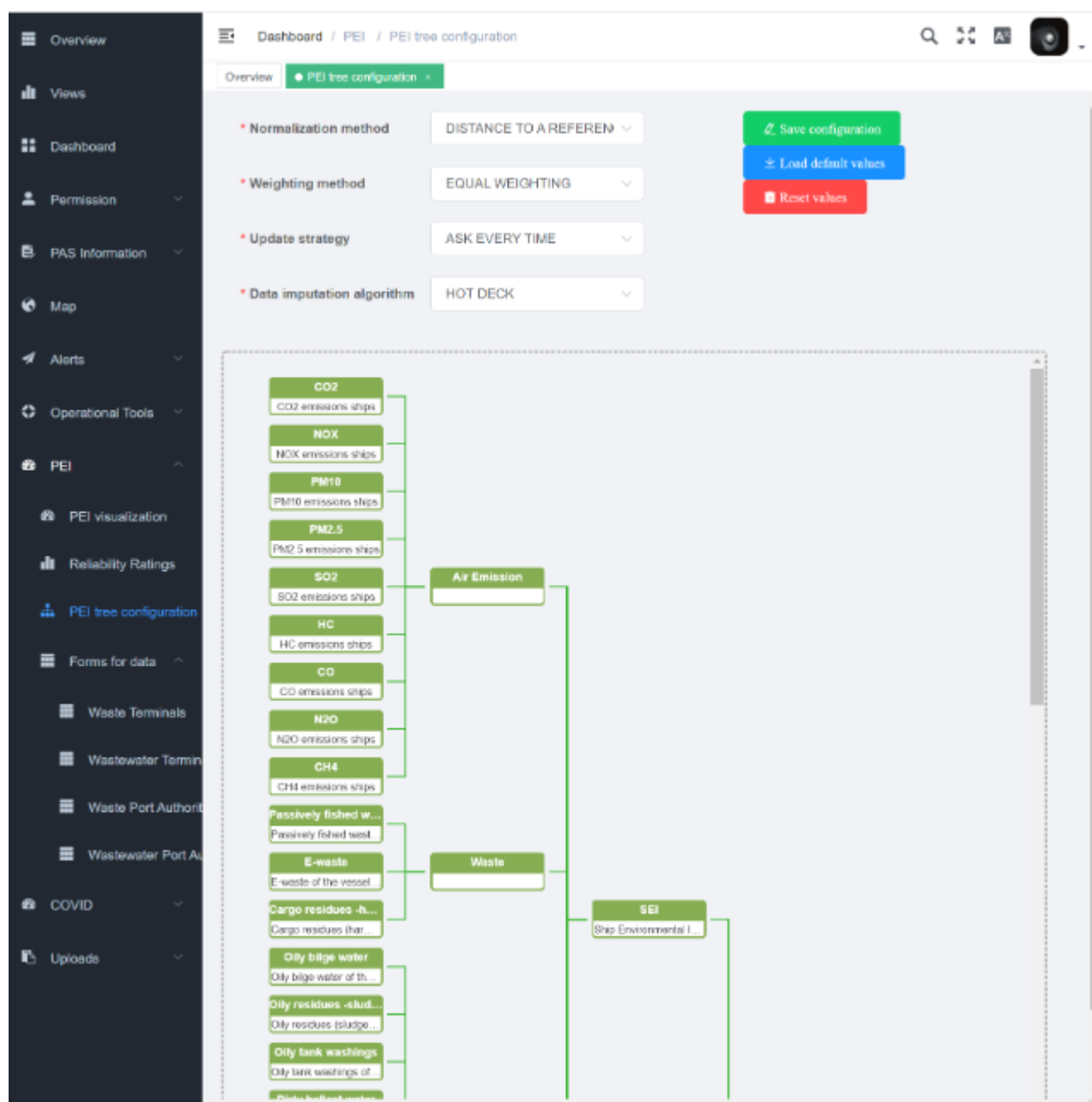
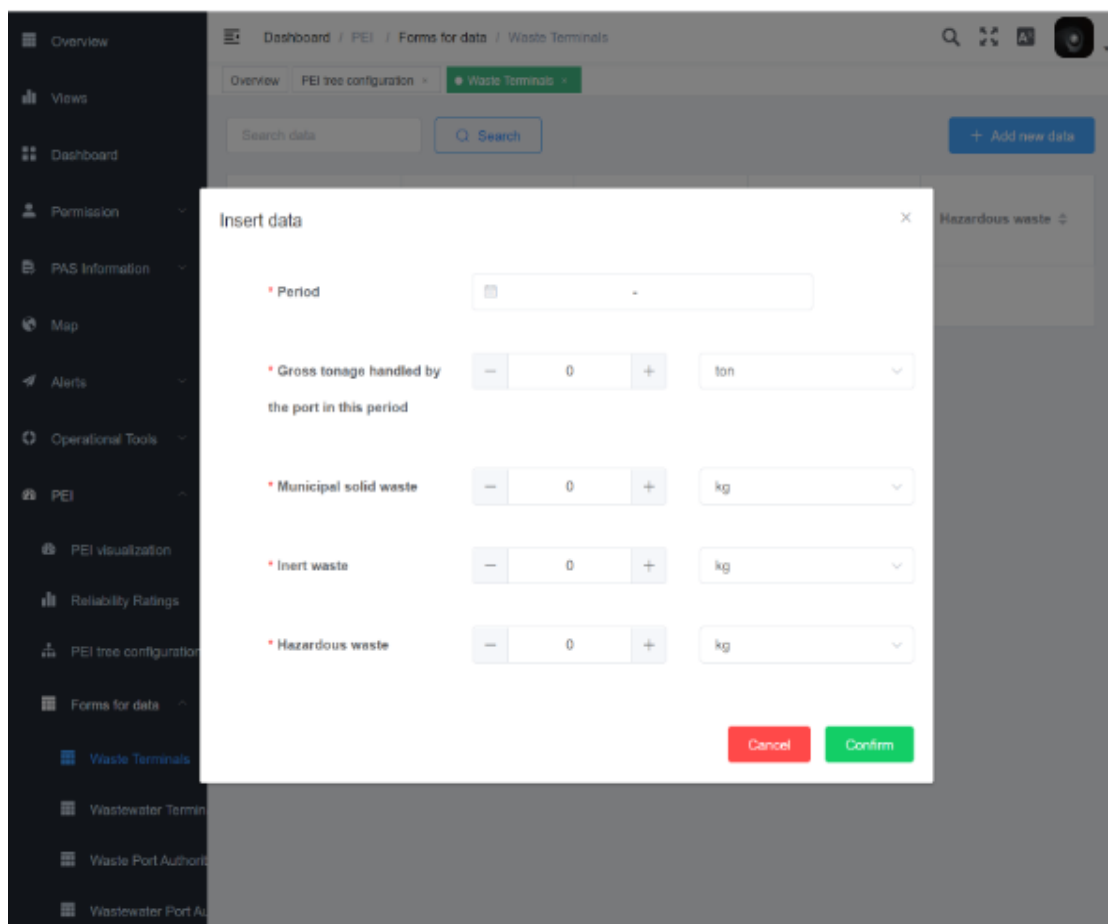


Figure 37. PEI Tree Configuration (default values)

Input eKPIs

This information is entered in the PIXEL platform by means of NGSI agents, able to convert from the specific format of each port to the homogenized FIWARE data model of KPIs, slightly adapted to PIXEL needs.

For the case of THPA, NGSI agents were developed and deployed, as will be described in the next section. However, considering that PIXEL targets small ports with limited resources, the PIXEL Dashboard allows to enter manually some data in web forms that will be properly imported in the IH. Currently the Dashboard supports the following data types: Waste Terminals, Wastewater Terminals, Waste Port Authority, Wastewater Port Authority. Below some screenshots are provided.



Dashboard / PEI / Forms for data / Waste Terminals

Overview PEI tree configuration Waste Terminals

Search data Search Add new data

Insert data

* Period

* Gross tonnage handled by the port in this period ton

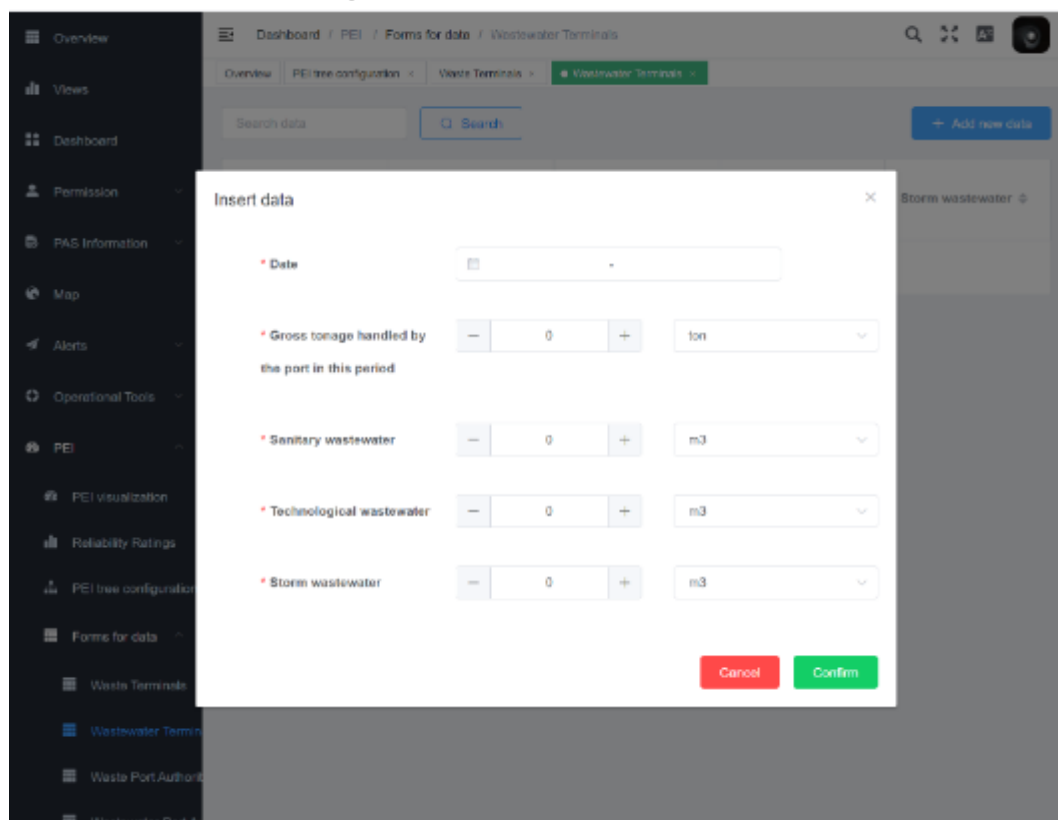
* Municipal solid waste kg

* Inert waste kg

* Hazardous waste kg

Cancel Confirm

Figure 38. PEI Forms. Waste Terminal



Dashboard / PEI / Forms for data / Wastewater Terminals

Overview PEI tree configuration Waste Terminals Wastewater Terminals

Search data Search Add new data

Insert data

* Date

* Gross tonnage handled by the port in this period ton

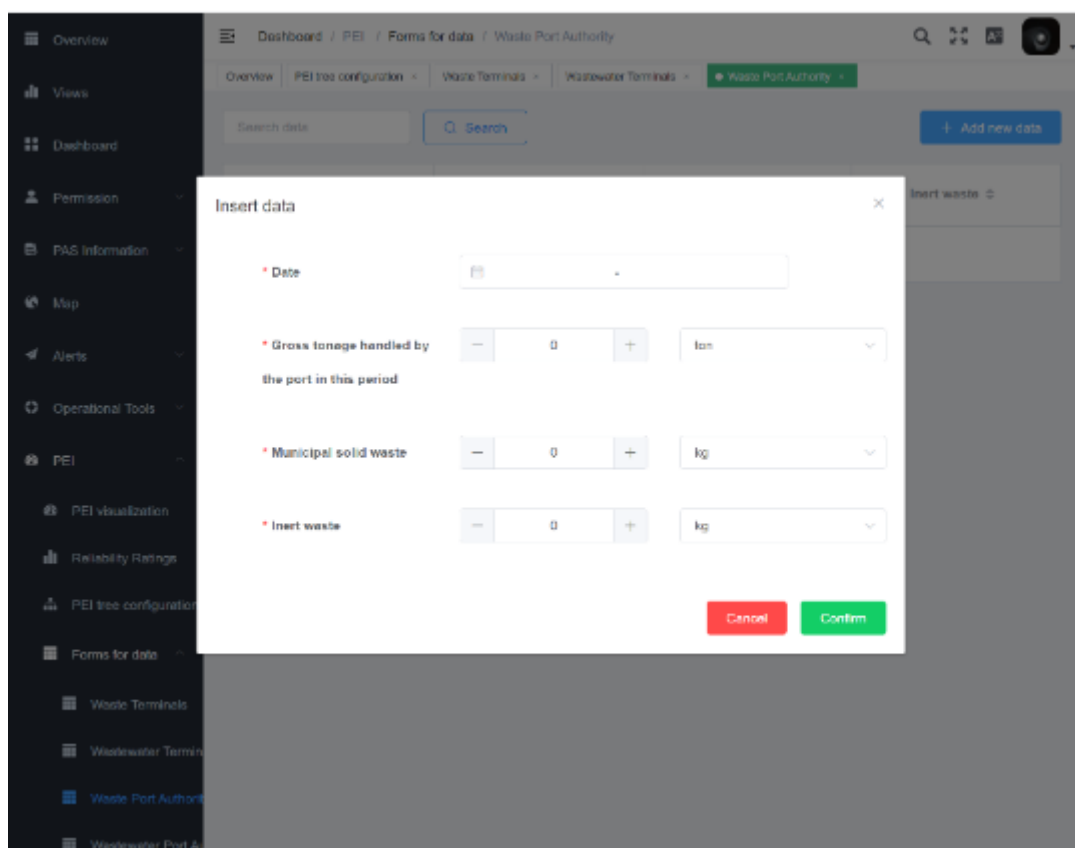
* Sanitary wastewater m3

* Technological wastewater m3

* Storm wastewater m3

Cancel Confirm

Figure 39. PEI Forms. Wastewater Terminal



Dashboard / PEI / Forms for data / Waste Port Authority

Overview PEI tree configuration Waste Terminals Wastewater Terminals Waste Port Authority

Search data Search Add new data

Insert data

* Date

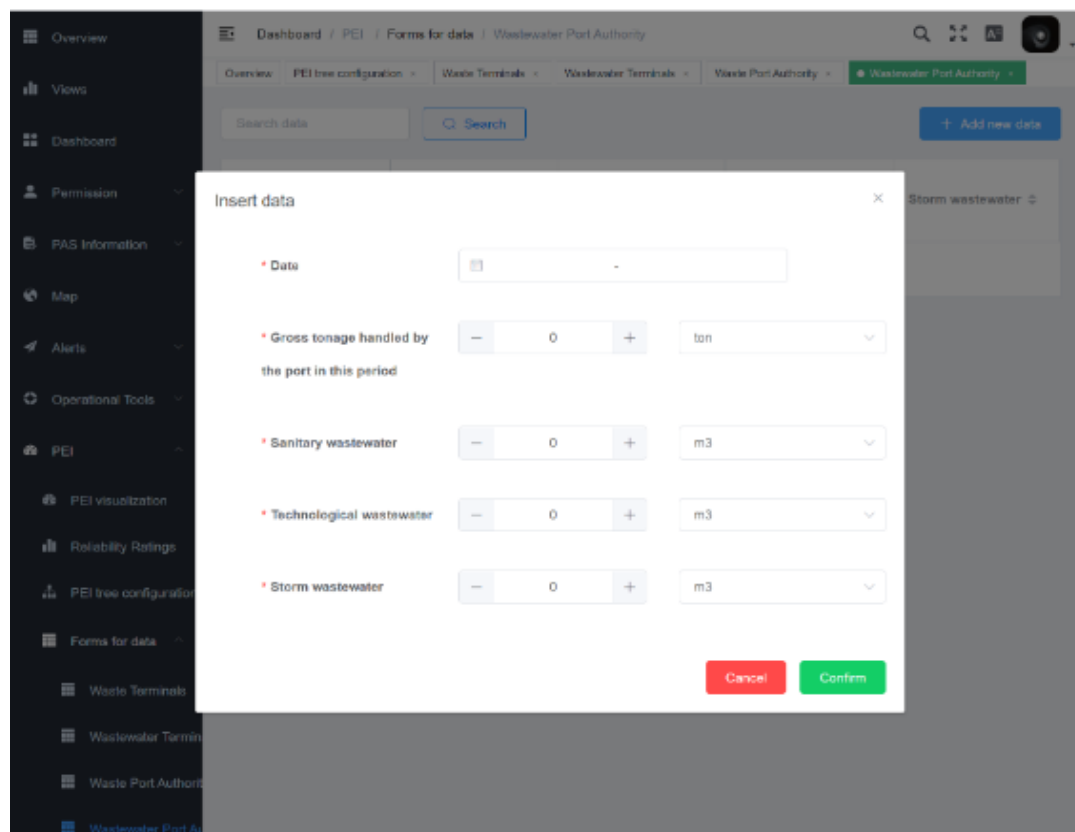
* Gross tonnage handled by the port in this period

* Municipal solid waste

* Inert waste

Cancel Confirm

Figure 40. PEI forms. Waste Port Authority



Dashboard / PEI / Forms for data / Wastewater Port Authority

Overview PEI tree configuration Waste Terminals Wastewater Terminals Waste Port Authority Wastewater Port Authority

Search data Search Add new data

Insert data

* Date

* Gross tonnage handled by the port in this period

* Sanitary wastewater

* Technological wastewater

* Storm wastewater

Cancel Confirm

Figure 41. PEI Forms. Wastewater Port Authority

Reliability Ratings

The Reliability Ratings is a way of indicating PEI readiness to IoT. Ideally all (or nearly all) information should flow in real time in the system, allowing a constant monitoring; however, the current situation in ports is far from a 100% digitalization status, and other ways of importing data needs to be used to calculate the PEI. Basically, different weights will be assigned depending on that, and therefore will be in the position of assessing a technology readiness index.

This functionality requires different input types:

- a. **Reliability rating tree:** this information, stored in the IH as a JSON file reflects the way the aggregation is performed for the different nodes of the tree. By default, an ARITHMETIC approach is used.
- b. **Reliability rating:** this information, stored in the IH as a set of different JSON files (one for each node in the tree) indicates the rating that is given to every node depending on how the data has been obtained. An example of a JSON file will be the following:

```
{
  "_index": "pei-reliability-rating",
  "_type": "_doc",
  "_id": "LM5X1nkB9zN0nbt0fvBP",
  "_version": 1,
  "_score": 0,
  "_source": {
    "dataPiece": "IMO number and gross tonnage of ships",
    "optimal": "realTimeAPI",
    "realTimeAPI": 1,
    "periodicAPI": 0.8056,
    "pixelProxyTool": 0.8611,
    "externalData": 0.75,
    "manualUploadToServer": 0.6111,
    "handtypedForms": 0.111,
    "averageValueFromLiterature": 0.2778
  }
}
```

As commented before, note that if there is a real time API, a weight of 1 is given, which will result in an ideal RR index. However, any deviation from this real time API incurs in a penalization factor (worst case for manual typing of data)

This information can be displayed from the PIXEL Dashboard in a more user-friendly way, as depicted below. For every piece of data, there is an optimal and a current retrieval way, with an independent and an aggregated rating (for the subindices and origin).



Data origin	Subindex	Piece of data	Optimal retrieval way	Current retrieval way	Reliability Rating	Aggregated RR (subindex)	Aggregated RR (origin)
Ships		IMO number and gross tonnage of ships	Real time API	Periodic API	80.56 %	67.96 %	83.47 %
	Air Emission	Main and auxiliary engine of ships	Periodic API	Average value from literature	36.84 %		
	Berth and maneuvering time of ships		Sensors	Periodic API	88.49 %		
	Waste	MARPOL annexes about waste	Pixel proxy tool	Periodic API	87.14 %		
	Wastewater	MARPOL annexes about wastewater	Pixel proxy tool	Periodic API	85.29 %	85.29 %	
Terminal	Air Emission	Emissions produced by terminal machinery	Sensors	Periodic API	73.47 %	73.47 %	85.42 %
	Waste	Waste produced by the terminals	Pixel proxy tool	Periodic API	87.37 %	87.37 %	
Global	Noise	Noise values Lden - Lnight - Leq	Sensors	Average value from literature	34.89 %	34.89 %	30.61 %
	Light pollution	Luminosity-lux	Sensors	Average value from literature	26.53 %	26.53 %	

Figure 42. PEI Reliability Ratings (Dashboard)

8.4. PEI deployment in PIXEL: Data used and NGSI agents developed

Once the configuration was clear. It was the moment to actually feed the platform with the data needed to run the PEI. For doing so, two actions must be secured: (i) ensuring the availability of the data according to the configurations set and (ii) proper introduction (following the aforementioned eKPI format) of that data into the platform via the deployment of specific NGSI agents.

As these two actions were rather different among ports, this section is sub-divided in four cases where evidence of their execution is included. It is worth to mention that the work was not completely decoupled. All agents followed the output eKPI format and the majority of them were re-used among ports (or, at least, part of them).

8.4.1. GPMB: Port of Bordeaux

An initial analysis was carried out considering the data and frequency provided by GPMB. Each row in the following table maps to an NGSI agent that was developed for running the PEI in the port:

Table 38. Analysis of the frequency of the data and the nature of PEI NGSI agents in GPMB

PEI NGSI Agent	Analysis of the agent (based on data availability and update frequency)
Air Emissions-SHIPS	Air emission data related to ships are obtained based on the vessel call (ETA and ETD are used to calculate berthing time and an average of 30 min is considered for manoeuvring time) and vessel type.
Waste and w. water- SHIPS	MARPOL data are directly obtained from the PMS (VigieSip) of GPMB. An agent has been developed to retrieve the data.

Air Emissions-TERMS	Data are directly obtained via the results of the PAS model. An agent has been designed to automatically retrieve the data. The input of this agent consists of the results (output) of the PAS, in particular of the document "pollutants_emissions.json".
Waste and w.water- TERMS	Data is sent via an excel file sent by mail on a monthly basis. The agent automatically retrieves data.
Noise- GLOBAL	Data is received every minute (HOPU station), but is aggregated per hour
Air Emissions- GLOBAL	Data is received every minute (HOPU station), but is aggregated per hour

Drawing from those premises, the agents were successfully developed. Here below a summary can be found aiming at representing their nature and results:

Table 39. Analysis of the frequency of the data and the nature of PEI NGSI agents in GPMB

PEI NGSI Agent Name (in repo)	Main indications and provisions. Comments, details, frequency, nature.
shipsAirEmission Link to the code repo:	<ul style="list-style-type: none"> Data inferred from vessel calls GPMB provides data 10 days after the vessel load/unload Will run effectively every week 9 KPIS =[ekpi-CO2-ships,ekpi-NOX-ships,ekpi-PM10-ships,ekpi-PM2.5-ships,ekpi-SO2-ship,ekpi-HC-ships,ekpi-CO-ships, ekpi-N2O-ships, ekpi-CH4-ships] Granularity at insertion in Orion/Elastic= inserts eKPIs every week
Marpol-ngsi Link to the code repo	<ul style="list-style-type: none"> [Data obtained from PMS] GPMB provides data on real time. 14 KPIS =[ekpi-e-waste-ships, ekpi-cargo-residues--harmful--ships, ekpi-passively-fished-waste-ships, ekpi-oily-bilge-water-ships, ekpi-oily-residues--sludge--ships, ekpi-oily-tank-washings-ships, ekpi-dirty-ballast-water-ships, ekpi-scale-and-sludge-from-tank-cleaning-ships, ekpi-other-oil-ships, ekpi-nls-type-x-ships, ekpi-nls-type-y-ships, ekpi-nls-type-z-ships, ekpi-nls-other-ships, ekpi-sewage-ships] Granularity at insertion in Orion/Elastic = inserts eKPIs every day
pas-peii	<p>It executes each hour and checks if a new execution of the PAS has taken place. Typically, the PAS will be scheduled weekly, however it can be run on-demand. Thus, this agent will always get the most recent execution of the PAS. If there is not a new one, the logic is then not applied and the agent remains in idle status.</p> <p>Afterwards, it proceeds to the calculation. The process is composed of two pieces: First, it analyses the value of each pollutant from the last PAS output. The records are expected to be time stamped on future dates (drawing from the nature of the PAS model, which aims at simulating port terminal activities forward), then some storage and consolidation must be performed. Here, the agent stores in a JSON file (DataExtracted.json) the forecasted values of each pollutant per day according to the last PAS simulation. On the following iterations, this file is consulted (day by day) in case the new predictions should update the pollution estimated for a certain date, overriding the entry in case the value is greater than the currently registered value.</p>

	Second, each week (actually, on Sundays - 22h30), it compiles the data of the past week (from Monday to Sunday) existing on DataExtracted.json, sums it up, normalises the resulting value to the total tonnage operated by the port during that week and obtains the value of all pollutants' eKPIS (units = kg/ton). Finally, the agent creates the NGSI-compliant entity and forwards it to the DAL (ORION Context Broker) to update each entity
marpolterminalwaste Link to the code repo	<ul style="list-style-type: none"> • GPMB provides data monthly • Will run every month • 3 eKPIS =[ekpi-municipal-solid-waste-terminal, ekpi-inert-waste-terminal, ekpi-hazardous-waste-terminal]
Noise Link to the code repo	<ul style="list-style-type: none"> • 2 eKPIS =[ekpi-noise-pollution-liden-global,ekpi-noise-pollution-lnight-global] • Granularity at insertion in Orion/Elastic= inserts eKPIs every day
Light Link to the code repo	<ul style="list-style-type: none"> • 1 eKPI =[ekpi-light-pollution-global] • Granularity at insertion in Orion/Elastic= inserts eKPIs every day

Conclusions of Agents development in GPMB:

All data was available and agents have been developed to gather data on a daily basis or whenever the data is pushed by the PMS of GPMB. Only the data related to waste water is manually provided.

Air emissions due to ships are based on the FAL forms (type of vessel, ETA and ETA, and considering an average manoeuvring time of 30 min). The use of tugboats is not taken into account since their use is not an available data via the PMS of GPMB. Air emissions due to terminal operations are obtained through simulation (not real and measured data) using the Port Activities Scenario (PAS) model. Noise and light pollution related data are obtained by only one sensor and can be improved. The data related with the use of the natural liquid gas dredged are not considered in the PEI.

8.4.2.APT: Port of Monfalcone

An initial analysis was carried out considering the data and frequency provided by ASPM (P12) and then APT (P16). Same as before, each row in the following table maps to an NGSI agent developed for APT's PEI:

Table 40. Analysis of the frequency of the data and the nature of PEI NGSI agents in APT

PEI NGSI Agent	Analysis of the agent (based on data availability and update frequency)
Air Emissions- <i>SHIPS</i>	The agent runs monthly and sends the info to the DAL.
Waste and w.water- <i>SHIPS</i>	The agent runs monthly and sends the info to the DAL.
Noise- <i>GLOBAL</i>	Data is received every minute (HOPU station)
Air Emissions- <i>GLOBAL</i>	Data is received every minute (HOPU station)

Drawing from those premises, the agents were successfully developed. Here below a summary can be found aiming at representing their nature and results:

Table 41. Analysis of the frequency of the data and the nature of PEI NGSI agents in APT

PEI NGSI Agent Name (in repo)	Main indications and provisions. Comments, details, frequency, nature.
Air Emission – SHIPS Link to the code repo	<ul style="list-style-type: none"> Vessel calls provided each month via an .xls file do not split between manoeuvring time and berthing time. Calculated pollutants have been properly approximated in order to deal with missing data. Runs once a month when the Excel file is uploaded to the agent via FTP. Historical data are available, 2019 has been defined as the starting year for the evaluation. 9 KPIS = [ekpi-CO2-ships,ekpi-NOX-ships,ekpi-PM10-ships,ekpi-PM2.5-ships,ekpi-SO2-ship,ekpi-HC-ships,ekpi-CO-ships, ekpi-N2O-ships, ekpi-CH4-ships]. Granularity at insertion in Orion/Elastic = inserts eKPIs every month
Waste and wastewater - SHIPS Link to the code repo	<ul style="list-style-type: none"> Data related to waste, wastewater and air pollution of each vessel reaching ASPM. Data is manually provided as Excel file each month by the personnel of ASPM Runs once a month when the Excel file is uploaded to the agent via FTP. Historical data are available since 01/01/2021- 14 KPIS =[ekpi-e-waste-ships, ekpi-cargo-residues--harmful--ships, ekpi-passively-fished-waste-ships, ekpi-oily-bilge-water-ships, ekpi-oily-residues--sludge--ships, ekpi-oily-tank-washings-ships, ekpi-dirty-ballast-water-ships, ekpi-scale-and-sludge-from-tank-cleaning-ships, ekpi-other-oil-ships, ekpi-nls-type-x-ships, ekpi-nls-type-y-ships, ekpi-nls-type-z-ships, ekpi-nls-other-ships, ekpi-sewage-ships] granularity at insertion in Orion/Elastic = inserts eKPIs every month-
Noise GLOBAL Link to the code repo	<ul style="list-style-type: none"> Data is provided by the HOPU sensor by means of MQTT packages No historical data Aggregation of noise data is done per day following the logarithmic mean formula. Real time data Two different pieces of data using the same data model (adapted to each) for submitting the eKPIs
Light GLOBAL Link to the code repo	<ul style="list-style-type: none"> Data is provided by the HOPU sensor by means of MQTT packages No historical data No aggregation of light related data is performed. Real time data

Conclusions of Agents development in APT

Some historical data are not available or limited to current year (regarding the data coming from MARPOL files, manually provided by port's operators and collected in the PMIS, APT started extracting and recording them in an .xls database from January 2021).

Current data was successfully imported into the PIXEL platform via the NGSI agents and is being continuously inserted on a monthly basis. Noise and light pollution related data are not aggregated (noise is calculated using the logarithmic mean per day). PEI in ASPM/APT is based, in addition to data provided by NSGI agents on several kinds of information heterogeneously collected and then processed by means of the PIXEL's dashboard and its input widgets.

8.4.3.THPA: Port of Thessaloniki

following table maps to an NGSI agent that was developed for running the PEI in the port:

Table 42. Analysis of the frequency of the data and the nature of PEI NGSI agents in THPA

PEI NGSI Agent	Analysis of the agent (based on data availability and update frequency)
Air Emissions-SHIPS	The agent may run weekly and store the data of the week or may be running continuously (querying the API) and sending the info to the DAL.
Waste and w.water- SHIPS	The agent must run quarterly (once every 3 months) and will encapsulate the data of waste per weeks (12 registries generated each time the agent runs)
Air Emissions-TERMS	The agent must run quarterly (once every 3 months) and will encapsulate the data per month (three registries to be generated each time the agent runs)
Waste and w.water- TERMS	The agent must run yearly (once per year) and will encapsulate the data per week (52 registries to be generated each time the agent runs)
Noise- GLOBAL	Data is received every minute (HOPU station), but will aggregate per hour
Air Emissions-GLOBAL	Data is received every minute (HOPU station), but will aggregate per hour

Drawing from those premises, the agents were successfully developed. Here below a summary can be found aiming at representing their nature and results:

Table 2. Analysis of the frequency of the data and the nature of PEI NGSI agents in THPA

PEI NGSI Agent Name (in repo)	Main indications and provisions. Comments, details, frequency, nature.
shipsAirEmission Link to the code repo DockerHub image: <i>pixelh2020/thpa-ngsi-shipsairemission</i>	<ul style="list-style-type: none"> Data ready for 2018-2021 THPA provides data (almost) in real time Will run effectively every week (even if run every hour at 2 AM (scheduled agent), it will not insert data until a week has passed from the last insertion) 9 KPIS =[ekpi-CO2-ships,ekpi-NOX-ships,ekpi-PM10-ships,ekpi-PM2.5-ships,ekpi-SO2-ship,ekpi-HC-ships,ekpi-CO-ships, ekpi-N2O-ships, ekpi-CH4-ships] Granularity at insertion in Orion/Elastic= inserts eKPIs every week
shipsWaste	<ul style="list-style-type: none"> Data ready for 2016-2020, but will insert only for 2018-2020

Link to the code repo DockerHub image: <i>pixelh2020/thpa-ngsi-shipsWaste</i>	<ul style="list-style-type: none"> • THPA provides data every 3 months • Uses THPA API from Container Terminal (CT) and Container Cargo Terminal (CCT) • Will run every day at 1 AM (scheduled agent) even if data is updated every 3 months • 14 eKPIs =[ekpi-e-waste-ships, ekpi-cargo-residues--harmful--ships, ekpi-passively-fished-waste-ships, ekpi-oily-bilge-water-ships, ekpi-oily-residues--sludge--ships, ekpi-oily-tank-washings-ships, ekpi-dirty-ballast-water-ships, ekpi-scale-and-sludge-from-tank-cleaning-ships, ekpi-other-oil-ships, ekpi-nls-type-x-ships, ekpi-nls-type-y-ships, ekpi-nls-type-z-ships, ekpi-nls-other-ships, ekpi-sewage-ships] • Granularity at insertion in Orion/Elastic = inserts eKPIs every month
terminalAirEmission Link to the code repo DockerHub image: <i>pixelh2020/thpa-ngsi-terminalairemission</i>	<ul style="list-style-type: none"> • Data ready for 2018-2021 • THPA provides data (almost) in real time • Will run effectively every week (even if run every hour at 2 AM (scheduled agent), it will not insert data until a week has passed from the last insertion) • 9 KPIs =[ekpi-CO2-ships,ekpi-NOX-ships,ekpi-PM10-ships,ekpi-PM2.5-ships,ekpi-SO2-ship,ekpi-HC-ships,ekpi-CO-ships, ekpi-N2O-ships, ekpi-CH4-ships] • Granularity at insertion in Orion/Elastic= inserts eKPIs every week
terminalWaste Link to the code repo DockerHub image: <i>pixelh2020/thpa-ngsi-terminalwaste</i>	<ul style="list-style-type: none"> • Ddata ready for 2018-2020 (not for 2016-2017) • THPA provides data yearly • Will run every day at midnight (scheduled agent) even if data is updated every year • 3 eKPIs =[ekpi-municipal-solid-waste-terminal, ekpi-inert-waste-terminal, ekpi-hazardous-waste-terminal] • Granularity at insertion in Orion/Elastic = inserts eKPIs every month, given a whole year
Noise Link to the code repo DockerHub image: <i>pixelh2020/thpa-ngsi-nise1820</i>	<ul style="list-style-type: none"> • For past data (2018, 2019, 2020): <ul style="list-style-type: none"> ○ Simulated from annual reports. ○ Data ready for 2018-2019-2020 ○ Will run only once, importing historical data through simulation (LABEL ng agent.type="manual") ○ 2 eKPIs =[ekpi-noise-pollution-ldn-global,ekpi-noise-pollution-night-global] ○ Granularity at insertion in Orion/Elastic= inserts eKPIs every day • For current (and future) data (from May 2021 on): <ul style="list-style-type: none"> ○ HOPU sensor. See explanation below (last row of this table).
Light Link to the code repo DockerHub image: <i>pixelh2020/thpa-ngsi-light1820</i>	<ul style="list-style-type: none"> • For past data (2018, 2019, 2020): <ul style="list-style-type: none"> ○ Based on lightpollutionmap.info - Wordl Atlas 2015 record ○ Will run only once, importing historical data through simulation (LABEL ng agent.type="manual") ○ 1 eKPI =[ekpi-lighte-pollution-global] ○ Granularity at insertion in Orion/Elastic= inserts eKPIs every day

	<ul style="list-style-type: none"> For current (and future) data (from May 2021 on): <ul style="list-style-type: none"> HOPU sensor. See explanation below (last row of this table).
<p>At the time of writing this deliverable real time NGSI agents for light and noise - considering the data sent by the HOPU stations - are currently under development but should be ready in short. There were hardware problems that required the station to be returned to fix the problem, thus incurring some delays.</p>	

Conclusions of Agents development in THPA

All data needed from 2018, 2019 and 2020 was gathered and imported. Some older data were available, but only partially and was not suitable to produce a reliable result. Current data is already being imported via the NGSI agents, but due to the disparity in the frequency update (from hourly to yearly) of some data from THPA, currently it is not recommendable to execute the PEI model for year 2021.

8.4.4.PPA: Port of Piraeus

An initial analysis was carried out considering the data and frequency provided by PPA. Same as before, each row in the following table maps to an NGSI agent developed for PPA's PEI:

Table 43. Analysis of the frequency of the data and the nature of PEI NGSI agents in PPA

PEI NGSI Agent	Analysis of the agent (based on data availability and update frequency)
Air Emissions-SHIPS	The agent may run daily and store the data of the previous day
Waste - SHIPS	The agent may run quarterly (once every 3 months)
Noise- GLOBAL	The agent may run daily and store the data of the previous day.

Drawing from those premises, the agents were successfully developed. Here below a summary can be found aiming at representing their nature and results:

Table 44. Analysis of the frequency of the data and the nature of PEI NGSI agents in APT

PEI NGSI Agent Name (in repo)	Main indications and provisions. Comments, details, frequency, nature.
Air Emission – SHIPS	<ul style="list-style-type: none"> Vessel calls agent used as the baseline data (its output for the agent). Raw data comes from the subscription to MarineTraffic API. Runs once a month when the Excel file is uploaded to the agent via FTP. Granularity at insertion in Orion/Elastic = inserts eKPIs every week.
Waste and wastewater - SHIPS	<ul style="list-style-type: none"> Data used is in .xlsx format that is uploaded to an FTP server. Historical data are available since mid-2019. Granularity at insertion in Orion/Elastic = inserts eKPIs every time there is a new document in the FTP

Noise GLOBAL	<ul style="list-style-type: none"> Data used is in .xlsx format that is uploaded to an FTP server. Historical data are available since mid-2019. Granularity at insertion in Orion/Elastic = inserts eKPIs every day
--------------	---

Conclusions of Agents development in PPA:

Several problems have been encountered, mainly technical. Those can be summarised as:

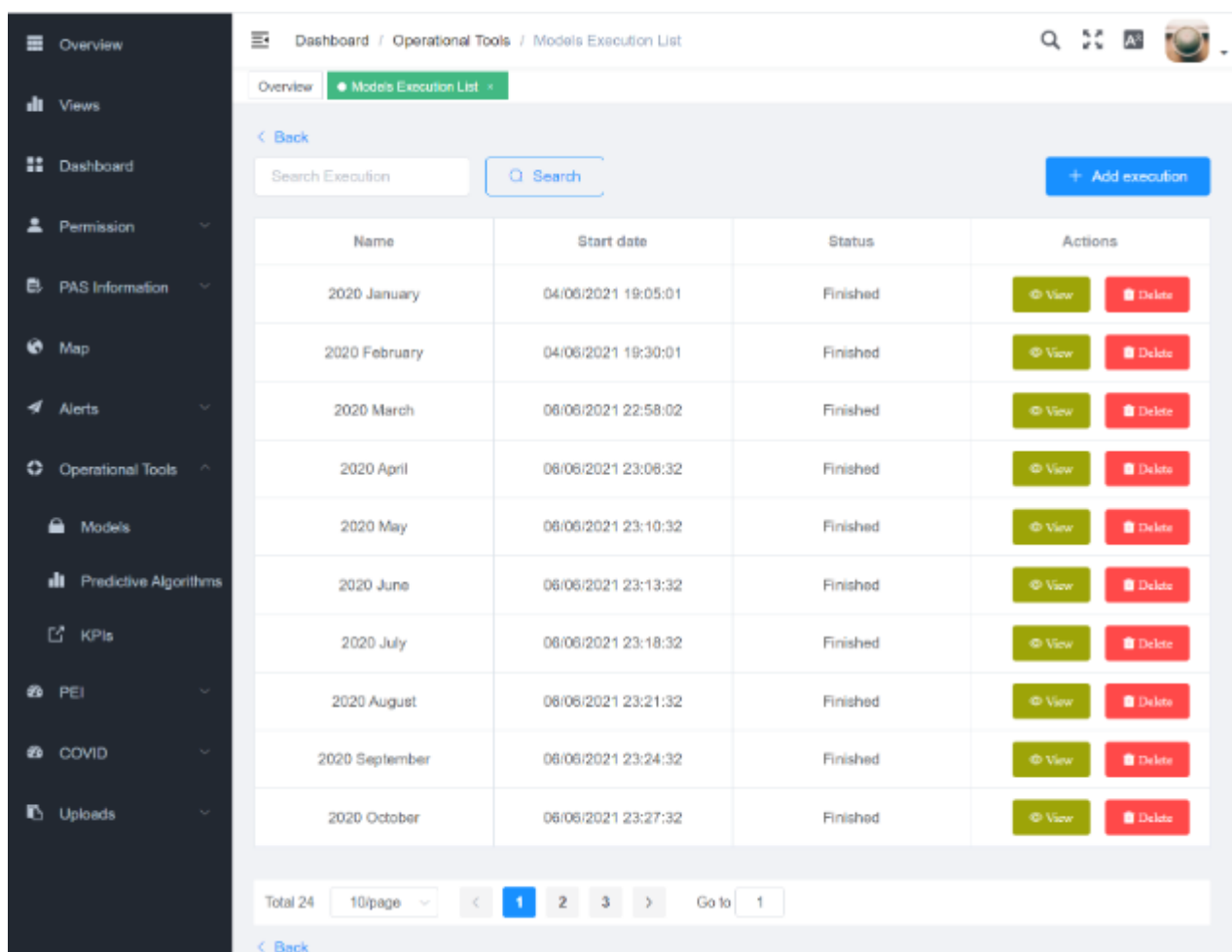
- Both MARPOL forms and the Noise station are provided in .xlsx files to an FTP server. The formats contained formulae and cell links, which made them hard to be managed via automated remote scripts.
- MarineTraffic subscription provides a series of endpoints which are limited in terms of queries/month and their structure was not ideal for PEI's purposes.

However, finally the agents were successfully deployed and the data has been made available in the IH of PPA's PIXEL instance.

8.5. Visualisation of results

The visualisation example of the model is shown below, based on the pilot in the port of Thessaloniki.

PEI executions can be launched (or scheduled) from the PIXEL Dashboard. The Figure below shows a list of PEI executions launched from the Operational Tools menu for the year 2020. In fact, a granularity of 1 month was set between every new PEI execution for any past year (for the case of THPA, this can differ from one port to another).



Name	Start date	Status	Actions
2020 January	04/06/2021 19:05:01	Finished	View Delete
2020 February	04/06/2021 19:30:01	Finished	View Delete
2020 March	06/06/2021 22:58:02	Finished	View Delete
2020 April	06/06/2021 23:06:32	Finished	View Delete
2020 May	06/06/2021 23:10:32	Finished	View Delete
2020 June	06/06/2021 23:13:32	Finished	View Delete
2020 July	06/06/2021 23:18:32	Finished	View Delete
2020 August	06/06/2021 23:21:32	Finished	View Delete
2020 September	06/06/2021 23:24:32	Finished	View Delete
2020 October	06/06/2021 23:27:32	Finished	View Delete

Figure 43. List of PEI executions in THPA (year 2020)

The monthly granularity allows retrieving values and displaying them graphically in a yearly visualization. Except for the executions of PEI models, which like any other model or predictive algorithm goes via the *Operational Tools Menu*, everything else (apart from the pure execution and scheduling of the model) related to PEI is managed through the *PEI Menu* entry. Clicking on the PEI Visualization submenu the port operator can visualize a series of results, as depicted below.

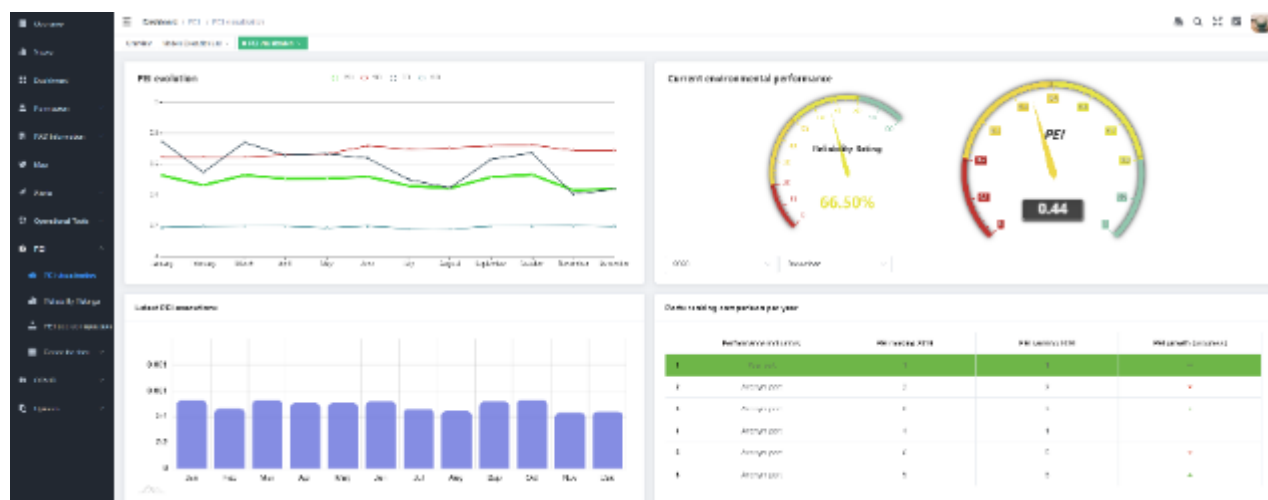


Figure 44. PEI results for THE- year 2020 (Part 1)

Basically, the following information can be visualised under different *spans* within the PEI visualization menu entry:

- **PEI evolution** A yearly graph (month by month) displaying the evolution of the PEI (in green) as the global index. Moreover, the 3 different sub-indices (SEI, TEI and GEI) are also displayed.
- **Current environmental performance:** A value for the PEI, in case port operators are just interested in a single value. In December 2020, THPA obtained a score of 0.44 (values are normalized), which gave the port room for improvement.
 - As it can be seen, the widget includes two SELECT items to choose the year and month.
 - Besides the PEI, there is also another value called *Reliability Rating* (RR) or IoT-Readiness level (IoTRL, see deliverable D5.3). It calculates the PEI readiness to a real IoT environment. A value of 65% is not bad as data has been somehow automatized through the agents; however, the long frequency update of some data impacts negatively on such value.
- **Latest PEI executions:** this is just another way of showing the PEI evolution in another format (bar chart), and without including sub-indices (SEI, TEI, GEI).
- **Ports ranking comparison per year.** This is a feature initially envisioned by the PEI and consists in the capacity of the system to allow the sharing of PEI results (under an anonymized framework) to enable a ranking comparison of the performance of all ports implementing the PEI.

Within the same visualization Dashboard, there is an additional widget as important as the previous ones, as it decomposes the PEI in environmental KPIs (eKPIs), which constitutes the groundings for the calculation of the PEI and allows an environmental manager to easily detect where and how much impact is causing to the environment. An example for 2020 is shown in the Figure below.

There the stakeholder can find all eKPIs that have been used by the PEI model, classified by ships, terminal, port authority and global. As the values are normalized, those eKPIs that are close to 1 mean that their impact on the environment is irrelevant and does not need urgent action; per contrary, those values that are close to 0 imply a significant impact on the environment and therefore, some strategy plan to transition to a greener status.

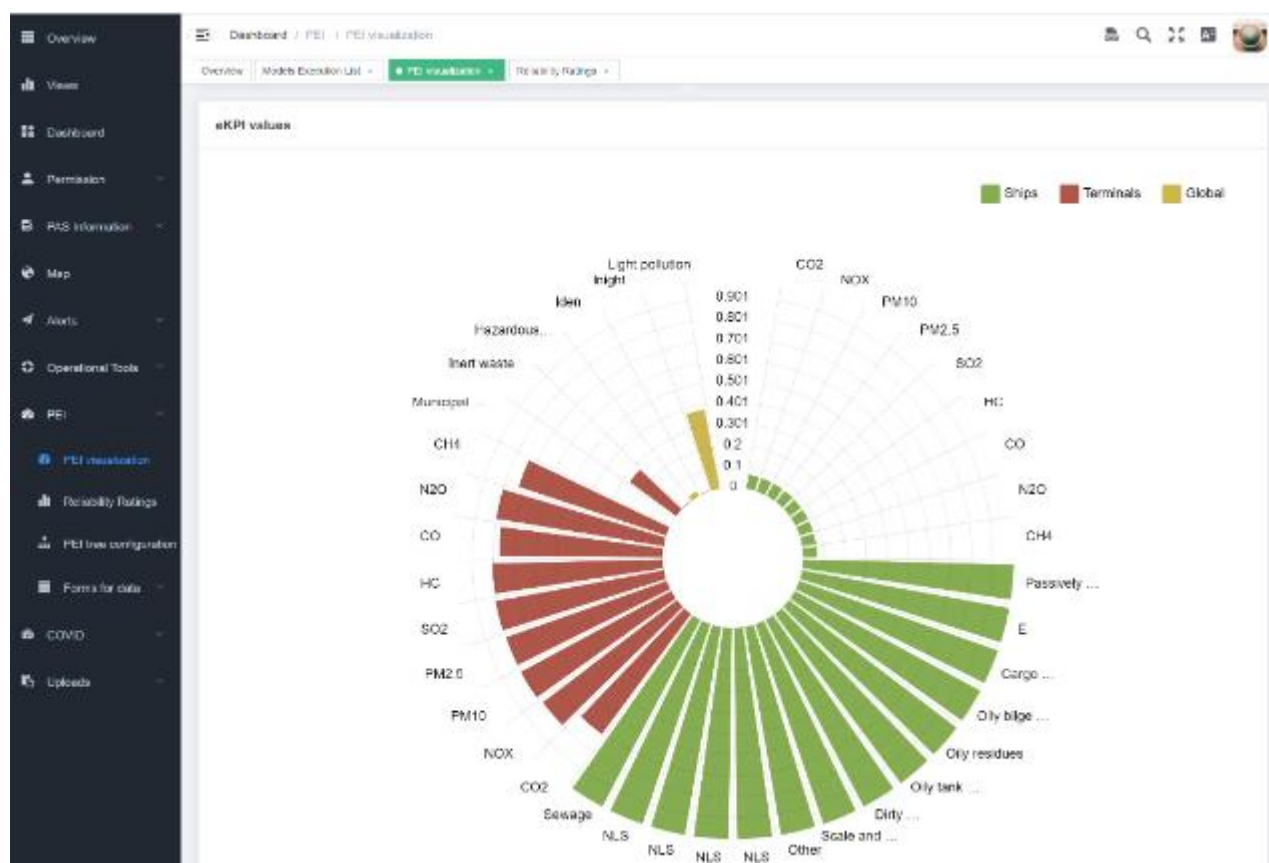


Figure 45. PEI results for the year 2020 (Part 2)

Besides, the visualization all this data can be exported to a document (PDF format) that can be used for internal monitoring reports, or even for external audits. This is one of the general advantages of having a centralized Information Hub able to store data as inputs (agents) and outputs (models): the user of the platform will be able to get them in various formats according to their needs.

The report document (See Annex F) highlights the following areas:

- **Calculation period:** month under study
- **Configurations:** normalization method, weighting method, aggregation method, update strategy, data imputation approach
- **Global values:** PEI value, RR value
- **PEI Index values:** SEI, TEI, GEI
- **eKPIs values:** CO₂, NO_x, PM₁₀, PM_{2.5}, SO₂, HC, CO, N₂O, CH₄, etc. (30+)
- **RR values:**

Recommendations: set of recommendations depending on the results and following the work done within WP5

8.6. Conclusions after technical deployment of PEI

PEI within PIXEL, as a technological functionality endorsed by an IoT platform, is ready (PEI model and agents) to support near real-time updates, but it is unlikely to happen (at least to its fullest extent) in the foreseeable future due to the existing gaps in the way data is treated in ports. Even if some data is managed in real time, plenty of data related to ships is not automatized, and it can take from weeks to months to have such information available; sometimes even typed manually.

It is expected that the transition to greener ports in general and the PEI model in particular will serve as main drivers to accelerate the real and almost complete digitalization of ports, as it is probably the most technological differentiator between small and big ports.

From the different outputs generated by the PEI for the different years and months, it is blatantly obvious that the ships are the most relevant area (negatively) contributing to the environmental impact, and this is where most of the overall port strategies should be focused.

Using the values of the PEI (results), especially analysing the evolution of it, ports are able to examine how to proceed to develop more environmentally friendly operations (thus improving the indicator). Finally, they have a tool to quantify the impact before and after implementing some specific strategy.

The sharing of the PEI results among ports is now partially supported by the PIXEL platform (due to the little number of ports implementing PEI, this functionality has not reached its technological maturity yet), even including anonymization, but further policies on how additional data (if needed) may be shared are still unclear and will depend on the analysis of each external port adopting PIXEL.

9. Conclusions

This document has explained the activities carried out in order to integrate the different components of the PIXEL platform and the development of the four use cases and the application of the Port environmental Index (PEI) in all of them.

The integration of the PIXEL platform components has been more costly than initially estimated because some integration considerations had not been addressed in previous technical work packages (WP4, WP5 and WP6). For this reason, before starting the integration, work has been done to define the integration mechanisms of the components that make up the platform (Information Hub, Operational Tools, Dashboard, Data Acquisition layer and Security) and to define a thorough validation methodology for testing them upon. In addition, during the course of the integration and the pilots, errors and shortcomings were found in the platform (usual in strongly innovative RIA projects) components that had not been detected until now.

In spite of the various drawbacks encountered, the platform was successfully integrated and ready in time for the development of the pilots. Even today, minor improvements are still being made to improve the usability of the platform and adapt it to the needs of the ports. As a matter of fact, the team considers that this will continue even after the finalisation of the project, as the usability requirements can always be refined and enhanced towards a better experience for the user.

At this point, all models planned initially to be tested in each pilot have been completed. In the deliverable “D7.3 - Pilots and Cross pilots collaboration report”, the last one of the WP7, more details can be found about the PIXEL architecture, models used in ports that were not initially planned, different opinions about using the PIXEL platform and lessons learned across all WP7 execution.

With regards to future actions, WP7 has put the means to allow the ports assess the impact of PIXEL solution on their activities.

AS a matter of fact, this deliverable has helped all pilots to reach the following status for all models that needed to be deployed in the ports:

- The PIXEL platform works.
- The model can be used through the PIXEL platform.
- The user can retrieve some results generated by the intended model.
- Those results consist of "delivered data" that is to be assessed.
- The personnel have been showcased on how to use the model and interpret those results.

Therefore, the work of WP7 is ended with this deliverable (altogether with D7.3). However, it must be noted that all WP7 technical team will be working to fine-tune potential mistakes or underperformances, including the improvement of usability to aid PIXEL Consortium reach as-good-as-possible KERs.

Appendix A– PIXEL Data Models

Data Models overview

The Data Models used in PIXEL follow the FIWARE standard NGSI. When it was possible the Smart Data Models provided by the FIWARE community were used :

<https://www.fiware.org/developers/smart-data-models/>

But when none of them were considered to fulfil the requirements, a new one was created, following FIWARE recommendations. These Data Models will be pushed to the FIWARE Community after validation of their use, in the PIXEL pilots.

Each Data Model is defined using a json schema that describes the entity type in Key Value format. The schema cannot be used to validate an entity provided with the NGSI Normalized format.

In PIXEL, the “source” property is used to track the origin of a data, while the URN to identify the source; but any valid source value could be accepted, with one technical limitation : A source could only provide one type of entity.

Data Models Management

All the Data Models used in PIXEL are stored in GIT :

- https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs

A version of this repository is deployed with the installation process in each port, in order to allow PIXEL components to rely on the models schema.

The FIWARE Data Models that are used, are copied in the PIXEL folder where all the Data Models created or adapted by PIXEL are stored. To allow the use of the Data Model schema as a full format specification, some existing FIWARE Data Models have to be extended, to add some properties specific to PIXEL.

In order to check the validity of the new Data Model, a docker container is available using node and mocha test framework to check the validity of each schema created for PIXEL.

- https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/validate

```
root@3dab3c9cab4:/app# npm run test
> schema-validator@1.0.0 test
> ./node_modules/.bin/mocha --recursive
Schema
Load Schema : https://fiware.github.io/data-models/common-schema.json
Load Schema : http://geojson.org/schema/Geometry.json
✓ /Pixel/VesselCall (3344ms)
✓ /Pixel/AirQualityObserved
✓ /Pixel/EnergyConsumptionMeasurement
✓ /Pixel/MarpolWaste
✓ /Pixel/Noise/NoiseLevelObserved
✓ /Pixel/Ping
✓ /Pixel/TideSensorObserved
Load Schema : https://fiware.github.io/data-models/specs/Weather/weather-schema.json
✓ /Pixel/Weather/WeatherObserved (854ms)
✓ /Pixel/WeatherObservedSencrop
✓ /Pixel/Traffic/TrafficFlowObserved
```

```

✓ /Pixel/MarpolTerminalWaste
✓ /Pixel/Traffic/TrafficFlow
✓ /Pixel/Weather/WindObserved
✓ /Pixel/KPI/EnvironmentalKeyPerformanceIndicator
Load Schema : https://smart-data-models.github.io/data-models/common-schema.json
✓ /Pixel/PhotovoltaicMeasurement (1256ms)
Load Schema : https://smart-data-models.github.io/dataModel.Device/device-schema.json
✓ /Device/Device (1256ms)
16 passing (7s)

```

Data Models List

The List of FIWARE Data Models used on PIXEL is presented below:

Table 45. FIWARE Data Models used on PIXEL list

Data Model name	Description
Device	An apparatus (hardware + software + firmware) intended to accomplish a particular task (sensing the environment, actuating, etc.).
OffStreetParking	Manage context about off street parking
TrafficFlowObserved	Manage context about traffic flow observation : averageVehicleSpeed, congested, vehicleType ...
WeatherObserved	An observation of weather conditions at a certain place and time. This data model has been developed in cooperation with mobile operators and the GSMA.

The list of PIXEL specific Data Model is presented below:

Table 46. PIXEL specific Data Model list

Data Model name	Status	Push to the community	Description
VesselCall	new model	In progress	Store VesselCall information on each port (arrival, departure, operation, cargo)
AirQualityObserved	update	No	Extension of the FIWARE model to store Bordeaux specific data
EnergyConsumptionMeasurement	new model	Too specific	Model to store Energy consumption provided by Bordeaux supplier.
MarpolWaste	new model	In progress	Store Marpol Waste information from the ship

MarpolWasteTerminal	new model	In progress	Store Marpol Waste information from the terminal
TideSensorObserved	new model	Too specific	Store Tide Sensor data form Bordeaux
WeatherObserved	update	No	Add of some specifics fields provided by the source
WeatherObservedSencrop	update	Too specific	Add of some specifics fields provided by the source
NoiseLevelObserved	update	No	Add of some specifics fields provided by the source
TrafficFlowObserved	update	No	Reduction of the FIWARE - Transportation / TrafficFlowObserved data model for the purpose of PIXEL project
TrafficFlow	new model	Too specific	Special model to store data specifics for one use case
WindObserved	update	To Discuss	Enhancement of the FIWARE - Weather / WeatherFlowObserved data model for the purpose of PIXEL project
EnvironmentalKeyPerformanceIndicator	new model	To Discuss	Adaptation of the FIWARE - KPI / KeyPerformanceIndicator data model for the purpose of PIXEL project
PhotovoltaicMeasurement	new model	To Discuss	The Data Model is intended to measure the continuous power transferred by the photovoltaic panel to an Inverter Device

Appendix B – HOPU Sensor Acquisition effort process

in this section the integration approach within the platform is presented. As an example, the HOPU station, which was purchased by the ports, is used.

Once the HOPU station is deployed on a site, it has two ways of sharing the data:

- Option 1. Cloud approach: This is the simplest approach for customers where all data from the HOPU stations (Smart Spot in the Figure below) are sent to the Homard cloud (provider), and the user can access via a web Dashboard (with credentials). If the data needs to be sent somewhere else, Homard includes the functionality to redirect the data via a custom IoT agent, using the NGSI Interface.

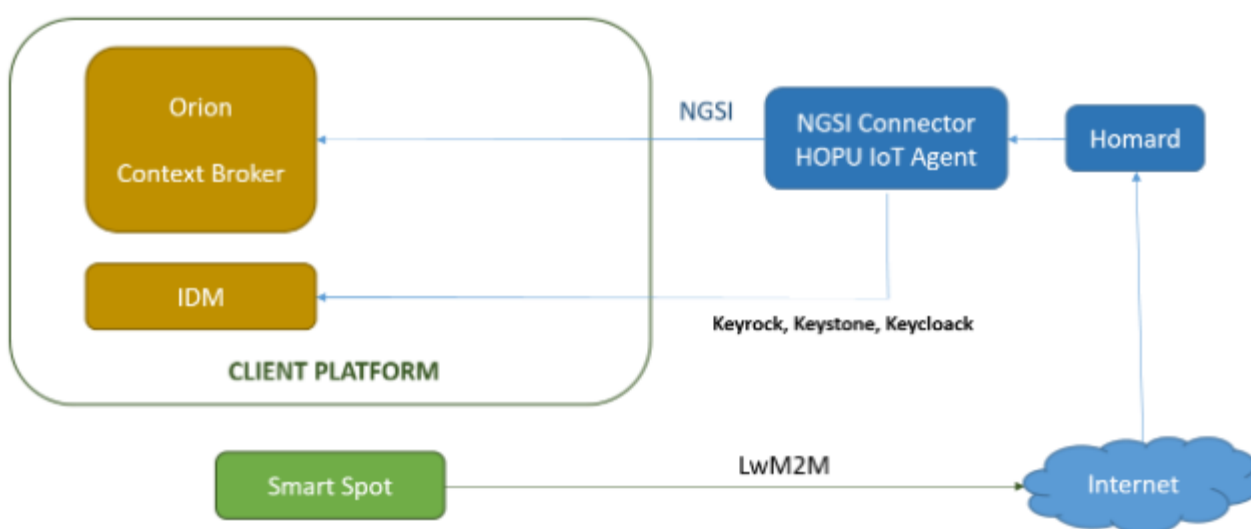


Figure 46. HOPU cloud approach

- Option 2. Local approach: the HOPU stations (Smart Spot) are able to send via MQTT all required information to a local Broker (e.g., Mosquitto). Therefore, the information can be treated locally without the need to request it from the Homard cloud.

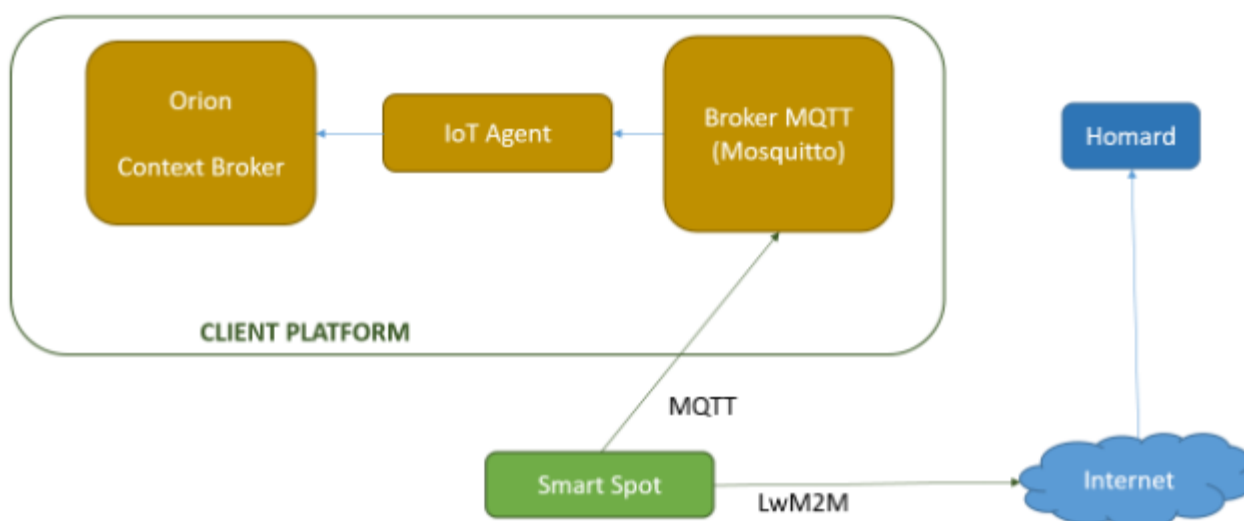


Figure 47. HOPU local approach

In the PIXEL project, not one single approach could be followed, due to the specific network configuration in the target ports. Therefore, the PIXEL platform is configured so as to be able to support both approaches.

The first approach (cloud approach) was used in THPA, and the needed adaptations are illustrated in the Figure below.

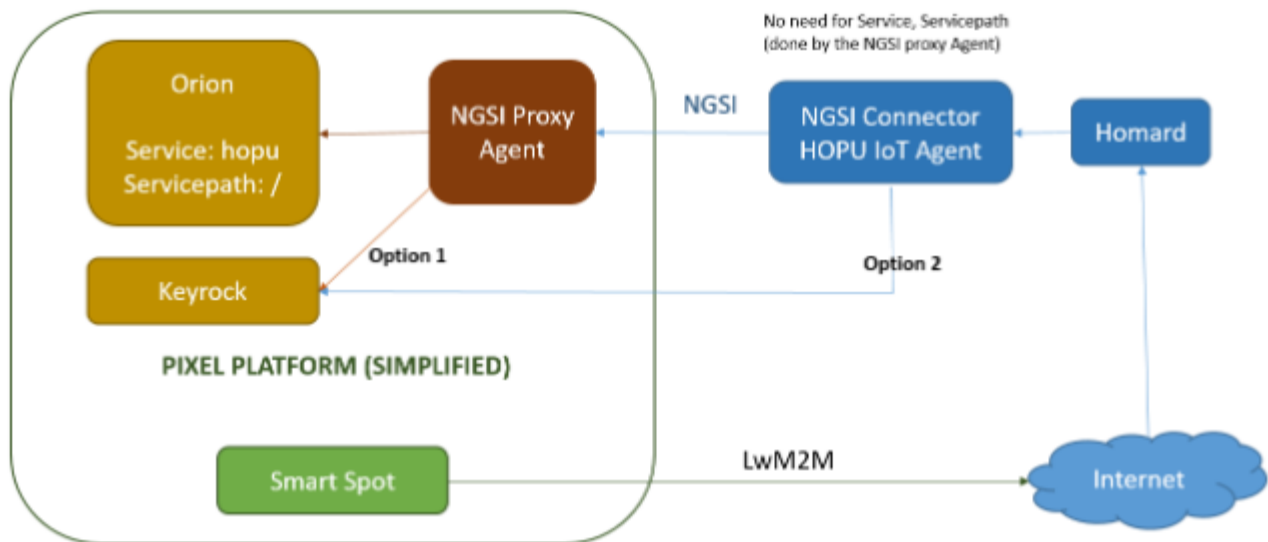


Figure 48. HOPU cloud approach adaptations for THPA

The PIXEL platform deployed an NGSI Proxy Agent to act as front end for the HOPU IoT Agent. Though this external IoT agent could use our Identity Manager, for simplicity purposes a permanent token for the requests was created. Moreover, in order to separate data, the Context Broker in PIXEL (Orion) uses a different Service and ServicePath.

Methods allowed by the NGSI agent were:

- **Version (GET)**

Example:

```
curl -H "X-Auth-Token: token_uuid" https://<endpoint>version
```

- **Entities (GET, PUT, POST, DELETE)**

Example:

```
curl -H "X-Auth-Token: token_uuid" https://<endpoint>/v2/entities
```

The last step is to tell the provider which minimal data should be sent back to the PIXEL platform in THPA. In our case, we are interested in noise and light values, which are sent periodically (e.g., every minute). The customer can check in the Homard Dashboard all available values, as depicted in the Figures below.

- **Noise (NoiseLevelObserved)**

- L_{Amax}
- L_{Amin}
- L_{Aeq}
- Units (dB)

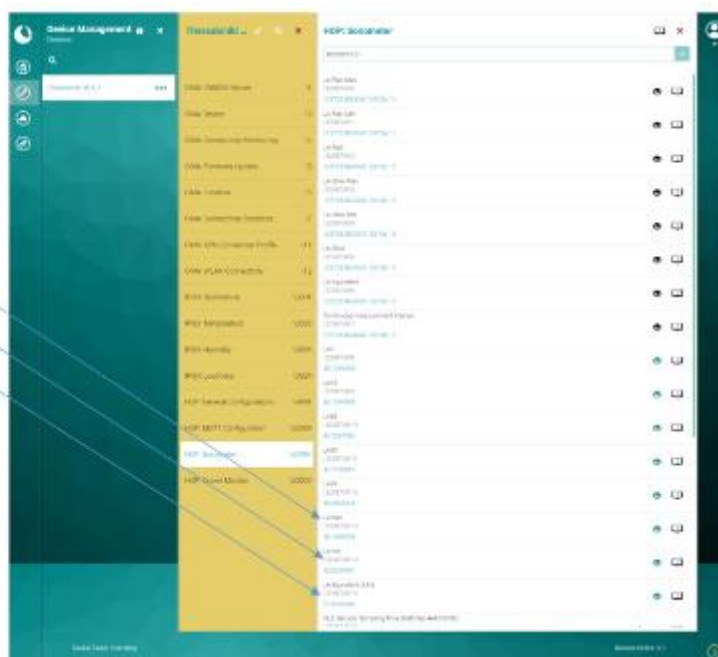


Figure 49. Needed noise values for THPA

- **IPSO: Illuminance**

- Max
- Min
- Units

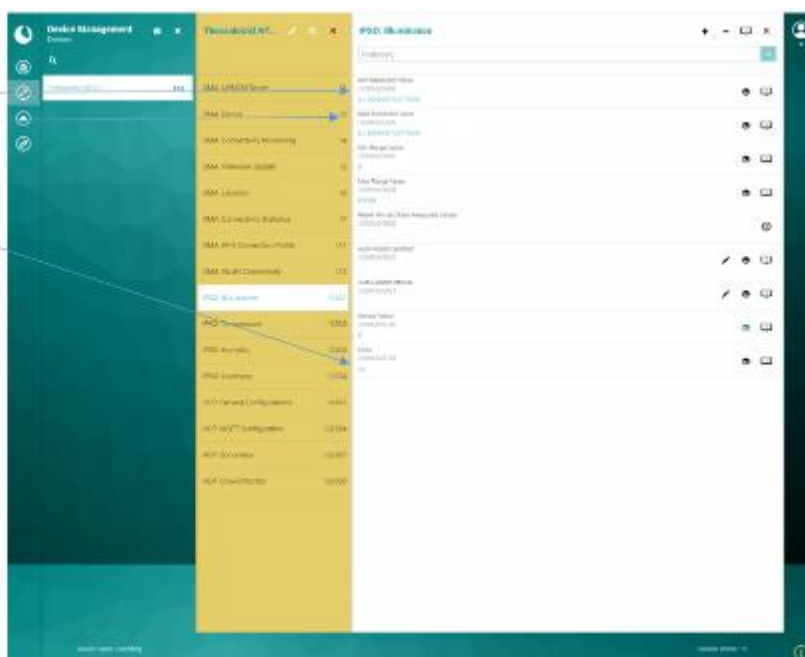


Figure 50. Needed light (illuminance) values for THPA

The second approach was followed by the pilots of Monfalcone and GPMB. In this case the information comes directly from the sensor without having to obtain the data from the Station cloud provider.

In this second approach, a Mosquitto MQTT server has been installed in the PIXEL platform to receive the data from sensors. Moreover, the NGSI Framework has been extended to retrieve data from MQTT. This framework is used by the agent in charge of storing the data of the sensor, into the PIXEL platform.

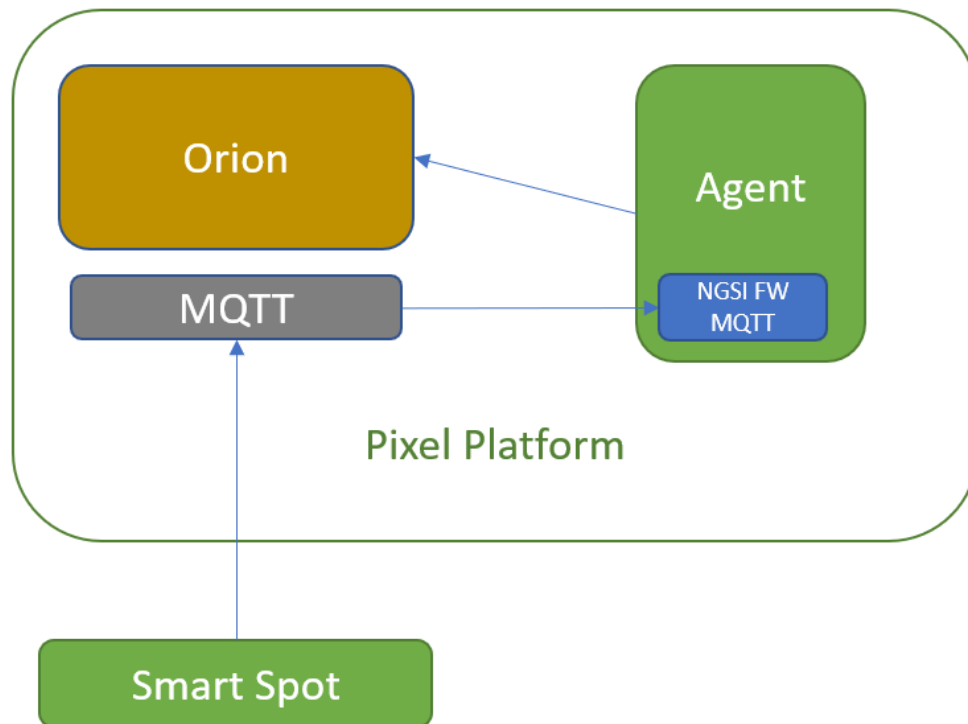


Figure 51. Direct integration through MQTT

Appendix C – NGSI Agents Cookbook

Requirements

In order to be deployed on a PIXEL Platform using the DAL and to be automatically detected by the Information Hub, NGSI Agents have to fulfil several requirements:

- Data Models should be registered
- URN to identify the Data Source should be defined
- Source ID provided with each entity, pushed in ORION
- DataSource have to be declared in Orion
- The agent should be dockerized with the needed labels
- The agent should be deployed with the Orchestrator

Note: A basic Ping NGSI agent as a practical case, will be included here. Code available at:

- <https://gitpixel.satrdlab.upv.es/benmomo/ngsi-agents-thpa/src/master/thpa-ping>

Data Models

The Data Models used by the NGSI Agents have to be added to the [PIXEL Data Model repository](#). The specific ones for PIXEL should be placed under the *specs/Pixel* folder.

The path of the json schema relative to spec folder will be needed, for example /Dummies/minimal-schema.json

For the Ping example the schema is at:

- https://gitpixel.satrdlab.upv.es/iglaub/Data_Models/src/master/specs/Pixel/Ping.

the schema.json (check for JSON syntax errors) must be validated, before deploying the agent, otherwise any syntax error will cause the DAL orchestrator to claim for (this can be seen by doing **docker logs dal_orchestrator**)

The data models repository must be updated on the platform:

- On PUBLIC host : git pull on /opt/pixel/Data_Models

```
root@vm-pixel-public:/opt/pixel/Data_Models# GIT_SSL_NO_VERIFY=false git pull
```

- On CORE host : /opt/pixel/Installation/docker/core/update-datarepo.sh

```
root@vm-pixel-core:/opt/pixel/Installation# GIT_SSL_NO_VERIFY=false git pull
root@vm-pixel-core:/opt/pixel/Installation/docker/core# ./update-datarepo.sh
```

Data Source URN

Each Data Source is identified with an URN following this syntax
urn:pixel:DataSource:<LOCODE>:<SourceName>

For example : urn:pixel:DataSource:farbod:TideSensorObserved

Provide Source ID in Entity

This source ID has to be provided in each entity pushed to Orion using the source field. IH uses this field to decide what to do with the entity.

Declare DataSource in Orion

In order to be automatically imported by Information Hub, 3 objects need to be created in Orion DataModel, SourceModelRelation and DataSource (always finished by DataSource).

Those objects are described in the Developer guide

If the agent using the DAL Orchestrator is deployed, let it manage this part.

One can also create those objects manually, or using the POST register api of the DAL Orchestrator

Dockerise NGSI Agents

In order to be identified and imported by the DAL Orchestrator the NGSI Agent has to fulfil several requirements using DOCKER LABEL :

Labels for all agents

- **ngsiagent.mode="insert"**: 'update' is the default mode and will provoke that the IH subscriber makes an update in Elastic. With 'insert', the IH subscriber makes an insert in Elasticsearch. The latter is the typical mode where an NGSI agent uses the same 'id' for every executions (no datetime timestamp), and the historic is then stored in Elastic
- **ngsiagent="pixel"**: this is the key label to be identified as a NGSI Agent
- **ngsiagent.type="daemon"**: define the type of NGSI Agent daemon, scheduled or manual
- **ngsiagent.datasources=["urn:pixel:DataSource:dummies"]**: this label provides the name of the datasource managed by this agent
- **ngsiagent.datamodels=["/Dummies/minimal-schema.json"]**: this label provides the path to each JSON Schema generated by the agent. The Data Models Path is the relative path to the specs folder of the Data Models repository. For example, for the data model TideSensorObserved the label should be set as :
nsgsiagent.datamodels=["/Pixel/TideSensorObserved/schema.json"]

Labels for daemon agents

- **ngsiagent.internal.port**: the port exposing the API, also has to be specified with EXPOSE
- **ngsi agent.internal.path**: the base path of the API configured in the agent
- **ngsi agent.external.path**:* the base path of the API configured in the proxy, to expose the agent

Labels for scheduled agents

- **ngsi agent.scheduled**: the frequency to run the agent (CRON format).

```
* * * * *
| | | | |
| | | | |
| | | | +---- Day of the Week   (range: 1-7, 1 standing for Monday)
| | | +----- Month of the Year (range: 1-12)
| | +----- Day of the Month   (range: 1-31)
| +----- Hour                 (range: 0-23)
+----- Minute                 (range: 0-59)
```

Examples

Daemon

```
FROM nginx
LABEL nsgsiagent="pixel"
LABEL nsgsiagent.type="daemon"
LABEL nsgsiagent.internal.port="80"
LABEL nsgsiagent.internal.path="/api"
LABEL nsgsiagent.external.path="/empire"
```

```

LABEL ngsiagent.datasources=["urn:pixel:DataSource:dummies"]
LABEL ngsiagent.datamodels=["/Dummies/minimal-schema.json"]
EXPOSE 80
ENV PIXEL=test
ENV MYTEST=pixel
RUN mkdir /usr/share/nginx/html/api
RUN echo "Execute order 66" > /usr/share/nginx/html/api/order
ENTRYPOINT ["nginx"]CMD ["-g", "daemon off;"]

```

Scheduled

```

FROM ubuntu
LABEL ngsiagent="pixel"
LABEL ngsiagent.type="scheduled"
LABEL ngsiagent.scheduled="* * * * *"
LABEL ngsiagent.datasources=["urn:pixel:DataSource:dummies"]
LABEL ngsiagent.datamodels=["/Dummies/minimal-schema.json"]
ENV PIXEL=test
ENV MYTEST=pixel
ENV SCHEDULED_DELAY=0
COPY docker_entrypoint.sh /docker_entrypoint.sh
RUN chmod u+rx /docker_entrypoint.sh
ENTRYPOINT ["/docker_entrypoint.sh"]

```

Manual

```

FROM ubuntu
LABEL ngsiagent="pixel"
LABEL ngsiagent.type="manual"
LABEL ngsiagent.datasources=["urn:pixel:DataSource:dummies"]
LABEL ngsiagent.datamodels=["/Dummies/minimal-schema.json"]
ENTRYPOINT ["/bin/bash"]CMD ["date"]

```

- Create the Agent using DAL Orchestrator

The API from the Orchestrator is protected with an API-KEY. Its value could be found in the secrets folder of the PUBLIC host.

```
root@vm-pixel-public:/opt/pixel/Installation/docker/public/secrets# nano dal.orchestrator.api.token
```

The API is not exposed outside the platform, it requires a direct request from the PUBLIC host, or to create an SSH tunnel to access its swagger:

```
ssh -i <keyfile> -L 127.0.0.1:8088:127.0.0.1:8080 root@xxx.xxx.xxx.xxx
```

Then the swagger can be accessed.

In order to be used with the orchestrator, the docker image should be available in the local docker repository (docker pull). For example, for a test ping NGSI agent:

```
root@vm-pixel-public:/opt/pixel/Installation# docker pull pixelh2020/pingtest:0.1
```

One can request the list of available NGSI Agents images already available on the host with an API call (**/images in Swagger**):

```

curl -H "X-Auth-Token: default" http://172.17.0.1:8888/api/images
[
  {
    "id": "sha256:620877b976447800bc7ce8672d6b688369b429ad77afba0968f20088c8daf8fd",
    "tag": "pixelh2020/frbdtidesensor:1.0.0"
  }
]

```

For the ping NGSI agent the response would be:

```

[
  {
    "id": "sha256:476573a6b213a4e16152dd9cd0ca8a62dbf06c8452e13c17f8bcf9e8e922becc",
    "tag": "pixelh2020/pingtest:0.1"
  }
]

```


- Get a template

When the image of the NGSI Agents is chosen, a template to create it (**/images/{id}/template in Swagger**) can be generated.

```
curl -H "X-Auth-Token: default" http://172.17.0.1:8888/api/images/sha256:620877b976447800bc7ce8672d6b688369b429ad77afba0968f20088c8daf8fd/template
{
  "name": "/?[a-zA-Z0-9_-]+",
  "image": "pixelh2020/farbodtidesensor:1.0.0",
  "type": "scheduled",
  "scheduled": "22 * * * *",
  "datasources": [
    {
      "urn:pixel:DataSource:farbod:TideSensorObserved"
    }
  ],
  "datamodels": [
    {
      "/Pixel/TideSensorObserved/schema.json"
    }
  ],
  "environment": [
    {
      "key": "PATH",
      "value": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
    },
    {
      "key": "NODE_VERSION",
      "value": "13.6.0"
    },
    {
      "key": "YARN_VERSION",
      "value": "1.21.1"
    },
    {
      "key": "NODE_TLS_REJECT_UNAUTHORIZED",
      "value": "0"
    },
    {
      "key": "ORION_URL",
      "value": "changeit"
    },
    {
      "key": "NAMI_AUTH_URL",
      "value": "https://nami.bordeaux-port.fr/?q=accueil"
    },
    {
      "key": "NAMI_URL",
      "value": "https://nami.bordeaux-port.fr/hauteurs"
    },
    {
      "key": "NAMI_LOGIN",
      "value": "changeit"
    },
    {
      "key": "NAMI_PASSWORD",
      "value": "changeit"
    },
    {
      "key": "FIWARE_SERVICE="
    },
    {
      "key": "FIWARE_SERVICE_PATH="
    }
  ]
}
```

For the ping example, the response would be:

```
{
  "name": "/?[a-zA-Z0-9_-]+",
  "image": "pixelh2020/pingtest:0.1",
  "type": "scheduled",
  "scheduled": "* * * * *",
  "datasources": [
    {
      "urn:pixel:DataSource:Ping"
    }
  ],
  "datamodels": [
    {
      "/Pixel/Ping/schema.json"
    }
  ],
  "mode": "insert",
  "environment": [
    {
      "key": "PATH",
      "value": "/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
    }
  ],
}
```

```
{
  "key": "LANG",
  "value": "C.UTF-8"
},
{
  "key": "GPG_KEY",
  "value": "E3FF2839C048B25C084DEBE9B26995E310250568"
},
{
  "key": "PYTHON_VERSION",
  "value": "3.8.6"
},
{
  "key": "PYTHON_PIP_VERSION",
  "value": "20.2.3"
},
{
  "key": "PYTHON_GET_PIP_URL",
  "value": "https://github.com/pypa/get-pip/raw/fa7dc83944936bf09a0e4cb5d5ec852c0d256599/get-pip.py"
},
{
  "key": "PYTHON_GET_PIP_SHA256",
  "value": "6e0bb0a2c2533361d7f297ed547237caf1b7507f197835974c0dd7eba998c53c"
},
{
  "key": "LC_ALL",
  "value": "C.UTF-8"
}
]
```

- Create the NGSI Agent (**POST /ngsiagent in Swagger**)

The name of the agent is changed (to the name of the container) and the parameters are adjusted or remain in default values. It should be assured that the name matches the given pattern.

```
curl -X POST -H "X-Auth-Token: default" http://172.17.0.1:8888/api/ngsiagent -d @- <<EOF
{
  "name": "/my-agent",
  "image": "pixelh2020/frbodtidesensor:1.0.0",
  "type": "scheduled",
  "scheduled": "22 * * * *",
  "datasources": [
    "urn:pixel:DataSource:farbod:TideSensorObserved"
  ],
  "datamodels": [
    "/Pixel/TideSensorObserved/schema.json"
  ],
  "environment": [
    {
      "key": "ORION_URL",
      "value": "http://172.17.0.1:1026"
    },
    {
      "key": "NAMI_LOGIN",
      "value": "mylogin"
    },
    {
      "key": "NAMI_PASSWORD",
      "value": "mypassword"
    }
  ]
}
EOF
```

For the ping example, the previous response from the template is used, while the name is changed to '/ping-test', including the ENV variables: ORION_HOST (private IP of CORE VM). ORION_PORT is not needed (1026 as default), as well as ORION_SERVICE and ORION_SERVICEPATH. These last two were initially considered to be **PIXEL** and **/GRSKG** respectively but should **not** be included here (nor considered in the NGSI implementation code).

```
{
  "name": "/pingtest",
  "image": "pixelh2020/pingtest:0.1",
  "type": "scheduled",
  "scheduled": "* * * * *",
  "datasources": [
    "urn:pixel:DataSource:Ping"
  ],
  "datamodels": [
```

```
"/Pixel/Ping/schema.json"
],
"mode": "insert",
"environment": [
  {
    "key": "PATH",
    "value": "/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  },
  {
    "key": "LANG",
    "value": "C.UTF-8"
  },
  {
    "key": "GPG_KEY",
    "value": "E3FF2839C048B25C084DEBE9B26995E310250568"
  },
  {
    "key": "PYTHON_VERSION",
    "value": "3.8.6"
  },
  {
    "key": "PYTHON_PIP_VERSION",
    "value": "20.2.3"
  },
  {
    "key": "PYTHON_GET_PIP_URL",
    "value": "https://github.com/pypa/get-pip/raw/fa7dc83944936bf09a0e4cb5d5ec852c0d256599/get-pip.py"
  },
  {
    "key": "PYTHON_GET_PIP_SHA256",
    "value": "6e0bb0a2c2533361d7f297ed547237caf1b7507f197835974c0dd7eba998c53c"
  },
  {
    "key": "LC_ALL",
    "value": "C.UTF-8"
  },
  {
    "key": "ORION_HOST",
    "value": "10.90.1.53"
  }
]
}
```

Given the fact that the procedures runs smoothly, the result, would be:

```
{
  "id": "93d876b1b34303754fc67030effcf54bae046a3af39b69b30a7cd486ca73d42"
}
```

Now, through a GET request to the **/ngsiagent**, one can get the available agents. For the ping test the response would be:

```
[
  {
    "id": "93d876b1b34303754fc67030effcf54bae046a3af39b69b30a7cd486ca73d42",
    "name": "/pingtest",
    "type": "scheduled",
    "state": "exited",
    "status": "Exited (0) 30 seconds ago"
  }
]
```

If you want to check whether the information is arriving in Orion, one can just use a **curl** command, from the console in the PUBLIC VM:

```
curl http://10.90.1.53:1026/v2/entities -s -S -H 'Accept: application/json'
```

For the ping example, the response would be:

```
[{"id":"Ping","type":"DataModel","schemaUrl":{"type":"string","value":"http://schemas.pixel.internal/Pixel/Ping/schema.json","metadata":{}}},{ "id":"urn:pixel:SourceModel:Ping:Ping","type":"SourceModelRelation","model":{"type":"Text","value":"Ping","metadata":{}},"source":{"type":"Text","value":"urn:pixel:DataSource:Ping","metadata":{}}},{ "id":"PingTest","type":"Ping","from":{"type":"Text","value":"93d876b1b343-172.42.1.3","metadata":{}},"source":{"type":"Text","value":"urn:pixel:DataSource:Ping","metadata":{}},"when":{"type":"Text","value":"2021-01-23T19:28:18+02:00","metadata":{}}},{ "id":"urn:pixel:DataSource:Ping","type":"DataSource","name":{"type":"Text","value":"urn:pixel:DataSource:Ping","metadata":{}},"same-id-mode":{"type":"Text","value":"insert","metadata":{}}}]
```

For a better visual output, one might install jq (**sudo apt-get install jq**) and then make the same query with this tool.

```
curl http://10.90.1.53:1026/v2/entities -s -S -H 'Accept: application/json' | jq
```

So, the outcome, would be:

```
[
  {
    "id": "Ping",
    "type": "DataModel",
    "schemaUrl": {
      "type": "string",
      "value": "http://schemas.pixel.internal/Pixel/Ping/schema.json",
      "metadata": {}
    }
  },
  {
    "id": "urn:pixel:SourceModel:Ping:Ping",
    "type": "SourceModelRelation",
    "model": {
      "type": "Text",
      "value": "Ping",
      "metadata": {}
    },
    "source": {
      "type": "Text",
      "value": "urn:pixel:DataSource:Ping",
      "metadata": {}
    }
  },
  {
    "id": "PingTest",
    "type": "Ping",
    "from": {
      "type": "Text",
      "value": "93d876b1b343-172.42.1.3",
      "metadata": {}
    },
    "source": {
      "type": "Text",
      "value": "urn:pixel:DataSource:Ping",
      "metadata": {}
    },
    "when": {
      "type": "Text",
      "value": "2021-01-23T19:28:18+02:00",
      "metadata": {}
    }
  },
  {
    "id": "urn:pixel:DataSource:Ping",
    "type": "DataSource",
    "name": {
      "type": "Text",
      "value": "urn:pixel:DataSource:Ping",
      "metadata": {}
    },
    "same-id-mode": {
      "type": "Text",
      "value": "insert",
      "metadata": {}
    }
  }
]
```

Note: It is important that the ‘id’ of the data entity (PingTest) is different from the ‘id’ of the Data Model entity (Ping). Otherwise, there may be two entities with the same id (possible for Orion unless the ‘type’ field is different) but might cause inconsistencies in some IH components.

The command can be executed every minute, so as to check that when field is being updated.

Following, one can verify whether this information is also getting to the IH. First the subscriptions to Orion:

```
curl http://10.90.1.53:1026/v2/subscriptions -s -S -H 'Accept: application/json' | jq
```

For the ping example, the response would be:

```
[
  {
    "id": "6006a31653cbffafa7cd7304",
    "description": "Information Hub subscription to DataSource notifications.",
    "status": "active",
    "subject": {
      "entities": [
        {
          "idPattern": ".*",
          "type": "DataSource"
        }
      ]
    },
    "condition": {
      "attrs": []
    }
  },
  "notification": {
    "timesSent": 4,
    "lastNotification": "2021-01-23T17:26:38.00Z",
    "attrs": [],
    "onlyChangedAttrs": false,
    "attrsFormat": "normalized",
    "http": {
      "url": "http://172.28.1.15:9009/DataSource"
    }
  },
  "lastSuccess": "2021-01-23T17:26:38.00Z",
  "lastSuccessCode": 200
},
  {
    "id": "6006ac5a53cbffafa7cd7305",
    "description": "Information Hub subscription to the source urn:pixel:DataSource:Ping.",
    "status": "active",
    "subject": {
      "entities": [
        {
          "idPattern": ".*"
        }
      ]
    },
    "condition": {
      "attrs": [],
      "expression": {
        "q": "source==urn:pixel:DataSource:Ping"
      }
    }
  },
  "notification": {
    "timesSent": 6205,
    "lastNotification": "2021-01-23T17:34:20.00Z",
    "attrs": [],
    "onlyChangedAttrs": false,
    "attrsFormat": "normalized",
    "http": {
      "url": "http://172.28.1.15:9009/urn%3Apixel%3ADataSource%3APing"
    }
  },
  "lastSuccess": "2021-01-23T17:34:20.00Z",
  "lastSuccessCode": 200
}
]
```

After that, a query to the IH API for the sources (from the CORE-VM) can be made:

```
curl http://172.25.1.17:8080/archivingSystem/extractor/v1/sources
```

and the result would be (Ping example):

```
[
  {
    "sourceId": "urn:pixel:DataSource:Ping",
    "sourceTypeId": "Ping",
    "indexName": "arh-lts-ping"
  }
]
```

Two general tutorial sessions have been conducted during the Code Camp (Valencia - February 2020) and the Technical meeting (Rijeka - April 2020) with the technical partners, in order to explain the technical aspects of the framework. In the end, most of the agents developed to integrate data sources into the Pixel pilots, used the framework. During the development of the agents, some minor updates to the FW have been added in order to support specific functionalities of the agents.

Appendix D – Ping scenario outcomes

Table 47. Deployment of the end-to-end scenario steps

#	Description	Comments
1	Ping Data model	Schema available in our internal GIT repository
2	Ping NGSI Agent	Source code available in our internal GIT repository Includes additionally examples using pyngsi as library or framework Class SinkOrion() does not include Fiware-Service, and Fiware-ServicePath as it is not used for the subscription (default values)
3	Ping Dockerfile (and Dockerhub)	Source code available in our internal GIT repository Dockerhub at pixelh2020/pingtest:0.1
4	Ping NGSI Agent tested locally (Orion)	Entity id fixed to 'PingTest' to have only one entity in Orion instead of multiple ones
5	Ping NGSI Agent deployed in PIXEL platform	Done, but this step kicked a change on the methodology as needed in Orion-IH subscription to support two modes: (insert, update)
6	Ping NGSI Agent documentation	Code includes README.md with further explanation, also pointing to NGSI cookbook
7	Ping Model definition	The Ping model counts the number of pings within a set of pings (e.g., the ones within a timeframe). Needed inputs defined. Examples of getInfo.json and instance.json (with forceinput) provided.
8	Ping Model implementation	Ping model written in Python, tested locally with sample files of instance.json (available with the code) and SSH tunnels to the PIXEL platform. Checked: (i) able to get input data from IH, (ii) able to get input data from forceinput, (iii) able to write output data in Elastic, (iv) able to log execution in Elastic (start, end error). (v) Scheduled instance also tested
9	Ping model Dockerfile (and Dockerhub)	Dockerhub at pixelh2020/pingcount:0.1
10	Ping model publication (OT API)	Tested locally on new OT implementation

11	OT needed update and documentation (getInfo.json)	<p>Tested locally.</p> <p>Source code available in our internal GIT repository</p> <p>Current OT documentation (Readthedocs) updated</p> <p>Documentation for OT framework (Readthedocs) added.</p> <p>Documentation (README.md) of PIXEL platform e2e with OT API added within the source code in the internal GIT repository</p> <p>JSON schemas added</p>
12	Ping model execution (OT API)	<p>Tested locally on new OT implementation</p> <p>OT instances in core_ih subnet, thus using IPs from Elastic and IH under this subnet</p>
13	OT needed update and documentation (instance.json)	<p>Tested locally, Docker to Dockerhub pushed (pixelh2020/ot:0.2), instances and scheduledInstances,</p> <p>Added API functionality to access to docker logs through the API for instances/scheduledInstances. Previous (automatic) docker prune facility has been removed (is shifted as API functionality, added), Added API functionality to show current version (/utils/version), Tested to publish an instance in the PIXEL platform through OT API, Code update in internal GIT repository, Current OT documentation (Readthedocs) updated, Documentation for OT framework (Readthedocs) added., Documentation (README.md) of PIXEL platform e2e with OT API added within the source code in the internal GIT repository, JSON schema added for instances, scheduledInstances, available in internal GIT repository</p>
14	Ping model published in PIXEL platform	getInfo.json available
15	Ping model executed in the PIXEL platform	instance_json available
16	Dashboard documentation	Manual showing usage of dashboard with screenshots and options
18	Visualization in Kibana	<p>Visualization for the incoming ping data of the NGSI agent added in the README file (GIT repo)</p> <p>Visualization for the model (pingcount) is the same in this case as for the NGSI agent, commented in the corresponding README file</p>
19	Visualization in Dashboard	Build a visualization for this result in the Dashboard
20	Final general test	<p>Double check:</p> <ul style="list-style-type: none"> • Technical coordinator follows doc to run the e2e scenario • Non-technical partner follows doc to run the 2e2 scenario

Appendix E – AERMOD worst-case scenario (THPA)

This document introduces the AERMOD proposal scenario and how the results change according to the emission data collected from the ships Air Emission Agent and the terminal Air Emission Agent.

The proposed scenario is intended to provide a general approach to a worst-case scenario in order to enhance protection of the port and the city (adopt strategies and take actions). Considering the daily port activity, the concentrations of the pollutants in ambient air can exceed the limit values; that is why, to outline the guidelines to follow in this situation, the AERMOD model can help to prevent the excess of the target values. Potentially it is possible to work with future predictions (if input data is available), but in this document we will work with past historical data (which is available in the PIXEL platform) in the port of Thessaloniki. Ideally, both scenarios are of interest: working with past data allows having maximum values to confirm whether some actions should have been taken; on the other hand, working with predictions allows anticipating potential harmful situations and employs proactive mitigation procedures.

AERMAP data

There are multiple ways to model emission sources in ports, and it is dependent on various criteria: complexity, focus on specific targets instead of the overall impact, proper characterization of areas and object (e.g., ships, vehicles) movement, etc.

A basic scenario to facilitate understanding is used as a starting point, which later on can be enhanced depending on the feedback of ports and how they prefer to focus on some aspects or others.

The scenario presented is composed of four areas, one for each dock, representing both impact of ships and terminal activity. This is the reason why the areas in the Figure below cover not only sea but also terminal zones.



Figure 52. Emission areas

The emission rates are taken from the two agents deployed in the port of Thessaloniki (ships Air Emission Agent, terminal Air Emission Agent):

- The shipsAirEmission agent runs effectively every week and inserts 9 eKPIs: [ekpi-CO2-ships, ekpi-NOX-ships, ekpi-PM10-ships, ekpi-PM2.5-ships, ekpi-SO2-ship, ekpi-HC-ships, ekpi-CO-ships, ekpi-N2O-ships, ekpi-CH4-ships]
- The terminalAirEmission agent runs effectively every 3 months and inserts 9 eKPIs: [ekpi-CO2-terminal, ekpi-NOX-terminal, ekpi-PM10-terminal, ekpi-PM2.5-terminal, ekpi-SO2-terminal, ekpi-HC-terminal, ekpi-CO-terminal, ekpi-N2O-terminal, ekpi-CH4-terminal]

They are able to collect historical and current data from the terminals and ships. As we are dealing with historical data, we will consider for the moment years 2018-2020. For these years, we can focus on one eKPI or another. We can check the worst case throughout one year, or all of them:

- Worst week for shipsAirEmission according to the value of one specific eKPI
- Worst month for terminalAirEmission according to the value of the same specific eKPI. If we want to work with the same units, we can divide the value by 4 (week unit)

Therefore, the agent outputs need to be filtered to provide the maximum average eKPI value for each pollutant, as shown in Table 1 and Table 2 (The values are converted from gram/week to gram/second which is the typical unit for modelling sources).

Table 48. Ships Air emissions rate

SHIPS	
Pollutant	Value (g/s)
SO ₂	0.0176
NO ₂	0.0217
PM ₁₀	0.0021
CO	0.0016

Table 49. Terminal Air emissions rate

TERMINAL	
Pollutant	Value (g/s)
SO ₂	0.2595
NO ₂	1.9665
PM ₁₀	0.0799
CO	0.3038

For the realization of the emissions rates for each area, the sum of the terminal and ships values by pollutant is done. The results are as shown in Table 3.

Table 50. Total Air emissions rate

TERMINAL + SHIPS	
Pollutant	Value (g/s)
SO ₂	0.0693
NO ₂	0.4970
PM ₁₀	0.0205
CO	0.0763

These are the values that need to be assigned to the areas. For simplicity, a homogeneous assignment will be made, but this could be later fine-tuned, if one area is more active than the other in the normal port activity.

Regarding the sources displayed in the KMZ map (previous image), these areas and the grid of receptors are created from the port terrain data. The most important data are the Domain coordinates, the Area sources, and the receptors.

Table 51. AERMAP data for THPA

PORT DATA		
Needed input	Explanation and options	Example
Enter Title:	Choosing title	THESS
Enter Domain coordinates (UTM, southwest corner X)	Needed for defining the Domain size (X coordinate of the lower left corner)	660192.0
Enter Domain coordinates (UTM, southwest corner Y):	Needed for defining the Domain size (Y coordinate of the lower left corner)	4498885.0
Enter Domain length along X axis [m]:	Needed for defining the Domain size	4000
Enter Domain height along Y axis [m]:	Needed for defining the Domain size	3000
Enter Domain UTM zone:	UTM zone of the chosen area	34
Add a new source (Y/N)?	this option allows the user to create sources	Y
Source type:	Choose the source type of the newly created source	AREA
Number of sources:	Defines how many new sources of the chosen type will be created	4
AREA SOURCE		
AREA 1		
Source group Name:	Name for chosen source group	PAREA
Input number of vertices for %s. area source:	For defining any area source, the user must enter the number of vertices (corners) the chosen area source has (minimum of 3)	4
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	661719.12
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500477.53
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	661986.13

Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500332.24
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662080.63
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500028.81
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	661674.61
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500035.04
Enter emission rate [g/s]:	Enter the emission rate in g/s for the area source. The scripts will then convert this value to $\text{g}/(\text{sm}^2)$	Depends on the pollutant
AREA 2		
Source group Name:	Name for chosen source group	PAREA
Input number of vertices for %s. area source:	For defining and area source, the user must enter the number of vertices (corners) the chosen area source has (minimum of 3)	4
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662116.86
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500333.91
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662336.25
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500268.73
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662492.71
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4499995.28
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662232.35
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500001.03

Enter emission rate [g/s]:	Enter the emission rate in g/s for the area source. The scripts will then convert this value to $\text{g}/(\text{sm}^2)$	Depends on the pollutant
AREA 3		
Source group Name:	Name for chosen source group	PAREA
Input number of vertices for %s. area source:	For defining any area source, the user must enter the number of vertices (corners) the chosen area source has (minimum of 3)	4
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662463.10
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500252.99
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662715.00
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500240.82
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662830.96
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500045.81
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662599.78
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500028.66
Enter emission rate [g/s]:	Enter the emission rate in g/s for the area source. The scripts will then convert this value to $\text{g}/(\text{sm}^2)$	Depends on the pollutant
AREA 4		
Source group Name:	Name for chosen source group	PAREA
Input number of vertices for %s. area source:	For defining any area source, the user must enter the number of vertices (corners) the chosen area source has (minimum of 3)	4
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662824.74

Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500224.73
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	663084.90
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500210.38
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	663189.75
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500022.83
Area Source Ni vertex's UTM X coordinates:	User needs to enter the X UTM of each of the vertices (corners)	662921.42
Area Source Ni vertex's UTM Y coordinates:	User needs to enter the Y UTM of each of the vertices (corners)	4500034.21
Enter emission rate [g/s]:	Enter the emission rate in g/s for the area source. The scripts will then convert this value to $\text{g}/(\text{sm}^2)$	Depends on the pollutant
RECEPTORS		
Enter number of X axis receptors	Choose how many receptor “points” (nodes) will be placed along the X axis	21
Enter number of Y axis receptors	Choose how many receptor “points” (nodes) will be placed along the Y axis	21
Enter receptor X coordinates (UTM, southwest corner)	For adding another receptor grid over the previously defined one must enter the coordinates for the new receptor domain. Enter the X UTM coordinates of the lower left corner	661595.12
Enter receptor Y coordinates (UTM, southwest corner)	Enter the Y UTM coordinates of the lower left corner of the new receptor grid	4500261.00
Enter receptor domain length along X axis	Defining the receptors grids length	40
Enter receptor domain length along Y axis	Defining the receptors grids length	35
Enter number of X axis receptors	Choose how many receptor “points” (nodes) will be placed along the X axis	21

Enter number of Y axis receptors	Choose how many receptor “points” (nodes) will be placed along the Y axis	21
----------------------------------	---	----

AERMET DATA

Weather variations, which take part in the severity of the environmental situation, should be considered. To represent our scenario, the 2018 weather data has been taken from the AERMOD FTP server.

These two files are reflected in our scenario as thessaloniki 2018 which is the upper air mixing data file and thessaloniki 2018 F that is the hourly surface data file.

Both files are download from the following FTP servers:

- <ftp://ftp.ncdc.noaa.gov/pub/data/noaa> (Hourly surface data)
- <ftp://ftp.ncdc.noaa.gov/pub/data/igra> (Upper Air Mixing data)

Table 52. AERMED data for THPA

PORT DATA		
Needed input	Explanation and options	Example
Enter Title:	Choose title/Name which will be used through the input files	THESS
UPPER AIR MIXING (STAGE 1)		
Enter Upper air mixing data input file Name:	Upper air mixing data file location/name, so the model's .exe can find and use the data file	thessaloniki2018AU.txt
File Type (6201FB/FSL):	upper air mixing data file location/name and the data file format which is usually FSL (forecast systems laboratory) The other format which is no longer in use is the TD-6201 format	FSL
Observation period start (YY/MM/DD):	When the observation period starts	2018/01/01
Observation period end (YY/MM/DD):	When the observation period ends	2018/12/31
Station latitude in decimal degrees (example: 42.75N):	Weather station latitude (where the upper air mixing measurements were taken)	40.520N
Station longitude in decimal degrees (example: 73.8W):	weather station longitude (where the upper air mixing measurements were taken)	22.970E

Time Offset from Greenwich [h]:	number of time zones the location is away from Greenwich Time. Areas west of Greenwich will have a positive time adjustment and areas east a negative	-2	
HOURLY SURFACE DATA (STAGE 1)			
Enter Hourly Surface data input file Name	hourly surface data file location/name, so the model's .exe can find and use the data file	thessaloniki 2018 SF	
File Type	Hourly surface data file format which is ISHD in most cases The other older formats are CD-144, SAMSON, HUSWO, and TD-3280	ISHD	
Enter the station identifier:	WMO weather station ID where the hourly surface measurements were taken	99999 (when bla)	
Station latitude in decimal degrees	Weather station latitude (where the hourly surface measurements were taken)	40.533N	
Station longitude in decimal degrees	weather station longitude (where the hourly surface measurements were taken)	22.967E	
Time Offset from Greenwich [h]	number of time zones the location is away from Greenwich Time. Areas west of Greenwich will have a positive time adjustment and areas east a negative	-2	
Station elevation [m]	Elevation from the means sea level of the station where the hourly surface measurements were taken	7.00	
SECTROS (STAGE 3)			
Wind measured at height [m]	Anemometer height (usually 10 m)	10	
Specify sector period:	See appendix	ANNUAL	
How many sectors would you like to add (Max 12)	See appendix	2	
Starting sector:	See appendix	90	270
Ending sector:	See appendix	270	90

The midday albedo:	See appendix	0.1400	0.2075
Daytime Bowen ratio:	See appendix	0.4500	1.6250
Surface roughness length [m]	See appendix	0.00010	1.00000

AERMOD

Table 53. AERMOD data for THPA

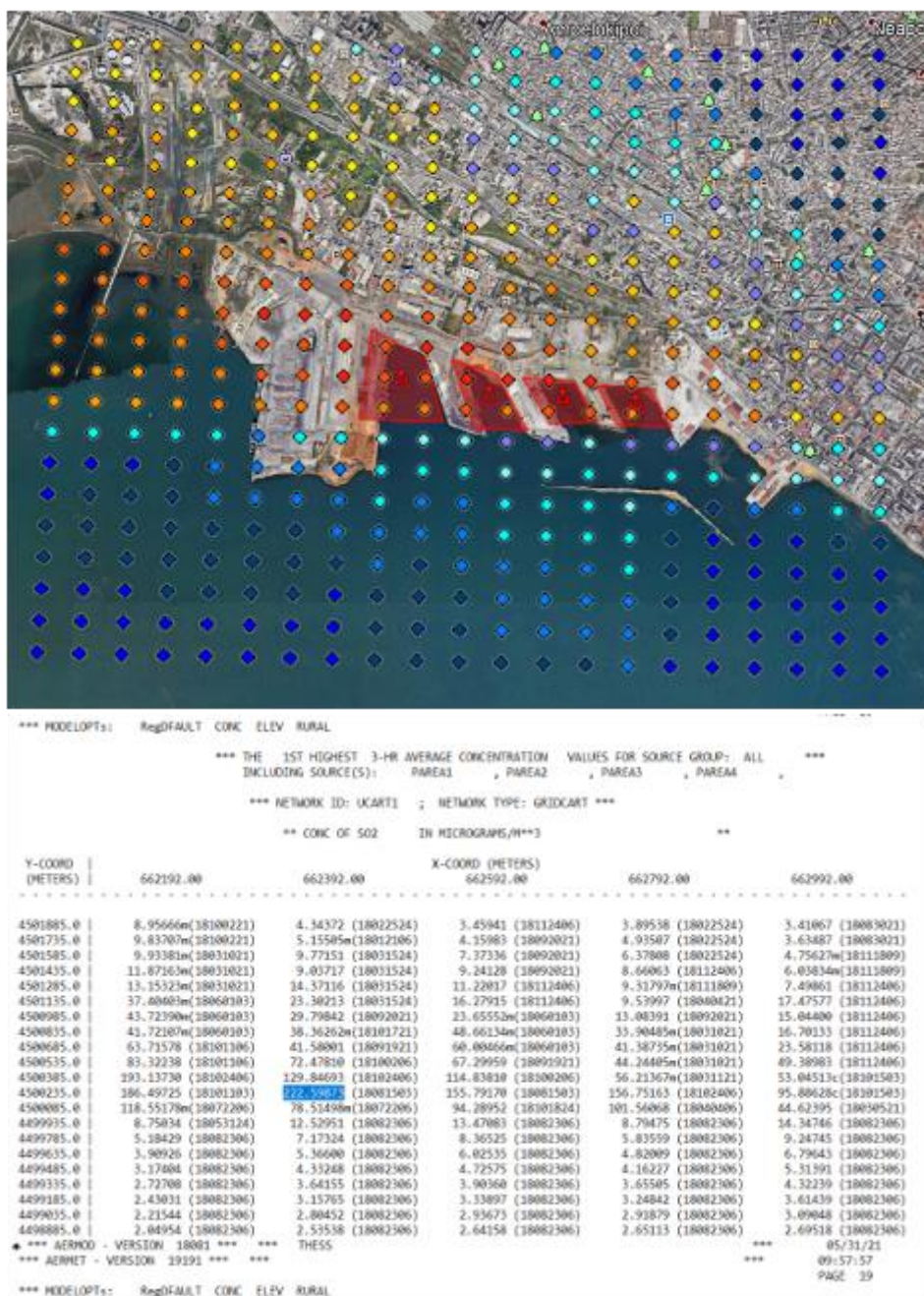
PORT DATA		
Needed input	Explanation and options	Example
Enter Title:	Title	THESS
Enter Pollutant's name:	Choose the pollutant for which you want to run the simulation	(SO ₂ , NO ₂ , PM ₁₀ or CO)
Enter base elevation [m]	Elevation of the main meteorological tower (where the surface data was collected)	10

RESULTS

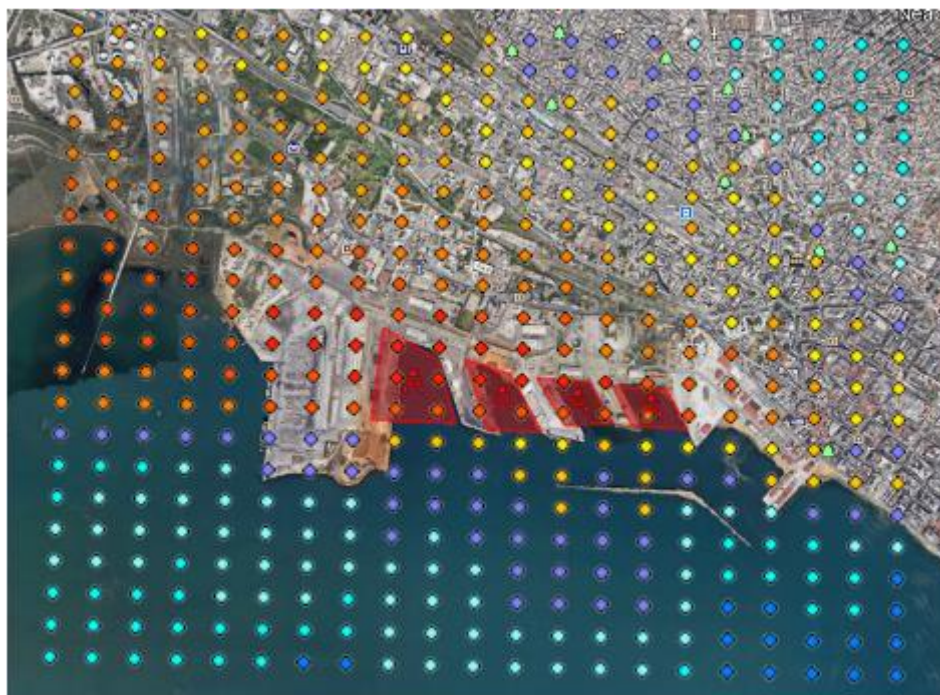
AERPLOT creates a map generated from the AERMOD output, where the data plotted should be the pollutant concentration in each of the receptors displayed and the source areas. The calculation made by AERPLOT is a 3 hour and a year pollutant concentration average in micrograms per m³.

The highlighted concentration in the table below is the highest one for the pollutant selected. It is important to consider the possibility of reaching the limit target value due to differences between the output time of 3-hour average in AERPLOT and the average time used in the ambient air quality standards for the European Union file.

S02



NO2



*** THE 1ST HIGHEST 3-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: ALL ***
 INCLUDING SOURCE(S): AREA1 , AREA2 , AREA3 , AREA4 ,

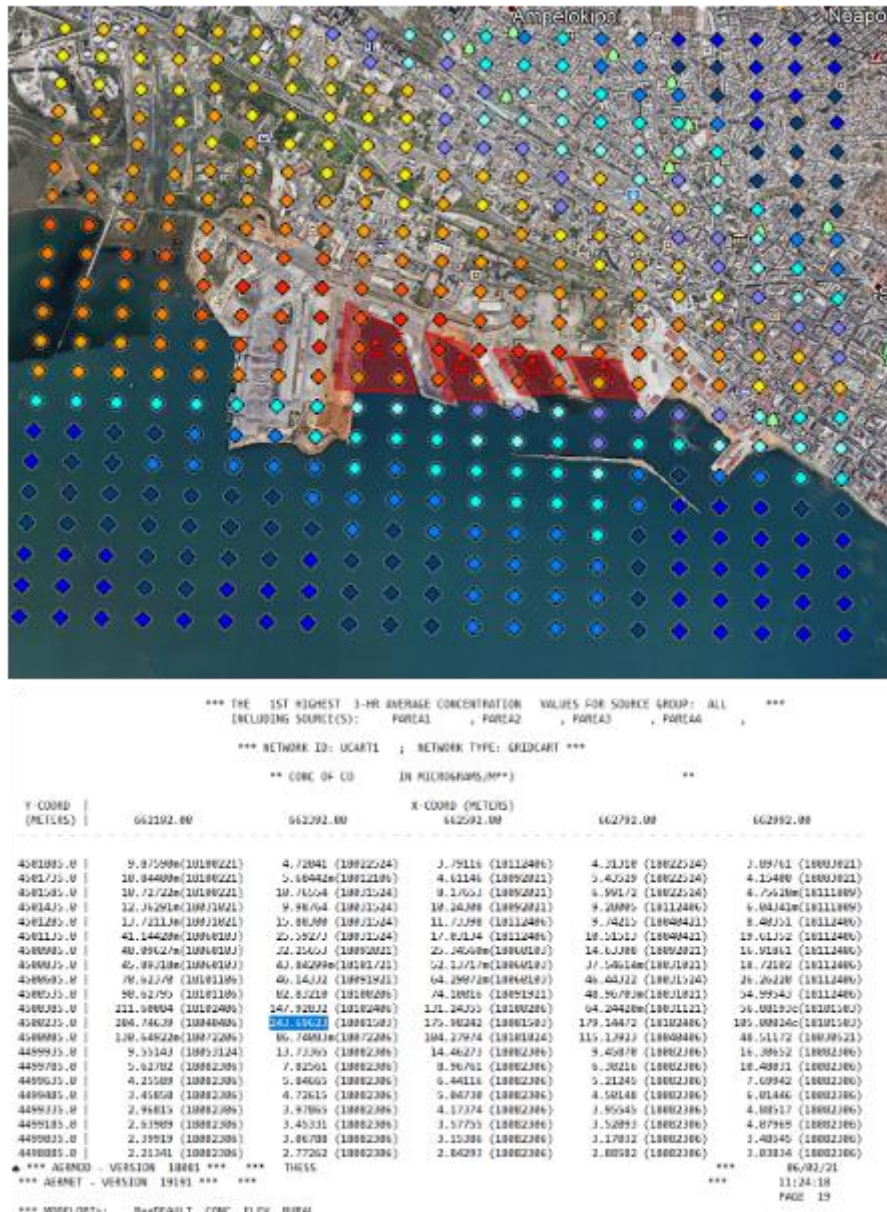
*** NETWORK ID: UCART1 ; NETWORK TYPE: GRIDCART ***

** CONC OF NO2 IN MICROGRAMS/M**3 **

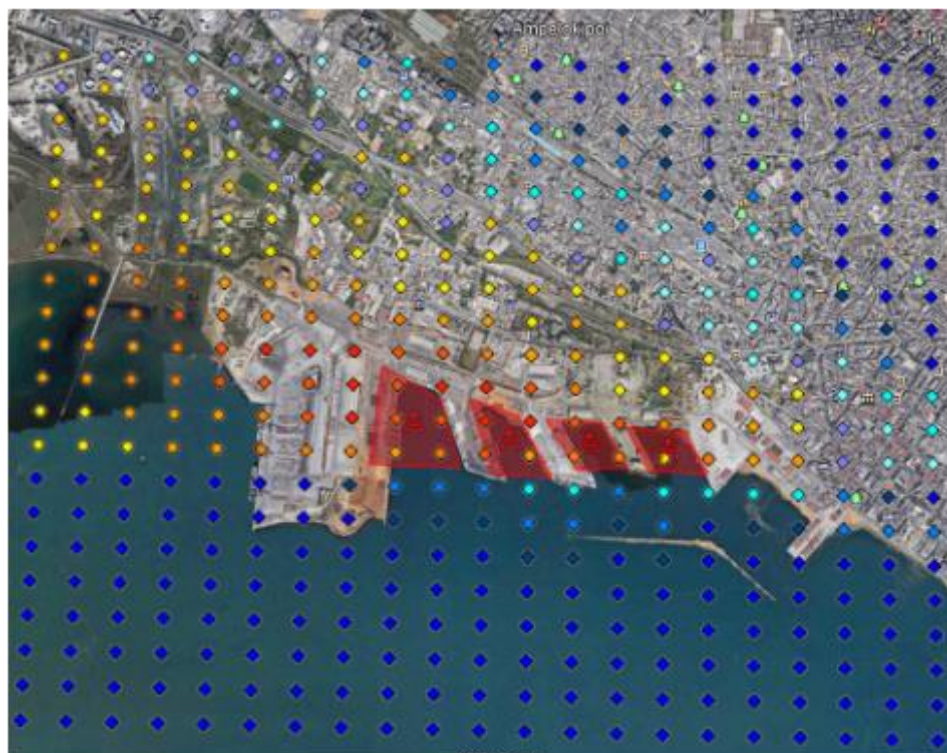
Y-COORD (METERS)	662192.00	662192.00	X-COORD (METERS)	662192.00	662192.00
4501685.0	63.81454m(18100221)	31.11350 (10022524)	24.55446 (18112406)	27.03121 (18022524)	20.84798 (10083021)
4501735.0	70.07661m(18100221)	26.75025m(10012106)	25.73724 (18091021)	35.14009 (18022524)	26.40207 (10083021)
4501585.0	69.29409m(18100221)	69.50484 (10031524)	52.71992 (18091021)	45.28009 (18022524)	31.70852m(18111009)
4501435.0	81.43605m(18031021)	64.40003 (10031524)	66.05606 (18091021)	60.42826 (18112406)	40.20770m(18111009)
4501285.0	90.33059m(18031021)	102.53944 (10031524)	77.19090 (18112406)	82.99206 (18080431)	56.30398 (18112406)
4501135.0	273.04719m(18000103)	105.61956 (10031524)	112.03787 (18112406)	67.95472 (18080431)	127.43452 (18112406)
4500985.0	219.18454m(18000103)	212.05027 (10092021)	105.59034m(18000103)	94.04501 (18092021)	109.65901 (18112406)
4500835.0	204.56302m(18000103)	279.49900m(18101721)	340.62956m(18000103)	242.21549m(18031021)	120.81986 (18112406)
4500685.0	455.43763 (18101106)	208.10790 (10091021)	420.03269m(18000103)	296.07551 (18031524)	160.20371 (18112406)
4500535.0	588.67256 (18101106)	528.05460 (18100206)	479.07094 (18091021)	315.95910m(18031021)	356.03984 (18112406)
4500385.0	1370.63317 (18102406)	804.15480 (18100206)	836.67762 (18100206)	809.55677m(18031121)	375.98623c(18101503)
4500235.0	1333.62706 (18101103)	1578.96149 (10081503)	1126.79286 (18081503)	1142.04756 (18102406)	684.54207c(18101503)
4500085.0	853.09718m(18072206)	560.30653m(18072206)	673.04764 (18101824)	736.22162 (18040406)	316.76072 (18030521)
4499935.0	62.05578 (18053124)	91.14740 (18082306)	94.67485 (18081306)	61.77193 (18082306)	104.10614 (18082306)
4499785.0	37.37095 (18082306)	51.03192 (18082306)	50.73093 (18081306)	41.16104 (18082306)	67.06881 (18082306)
4499635.0	28.25708 (18082306)	38.78833 (18082306)	42.18723 (18081306)	34.02698 (18082306)	49.31329 (18082306)
4499485.0	22.96228 (18082306)	31.35906 (18082306)	33.05837 (18081306)	29.35325 (18082306)	30.54371 (18082306)
4499335.0	19.70702 (18082306)	26.36810 (18082306)	27.33819 (18081306)	25.76125 (18082306)	31.31458 (18082306)
4499185.0	17.52359 (18082306)	22.07477 (18082306)	23.43454 (18081306)	22.90403 (18082306)	26.15102 (18082306)
4499035.0	15.91199 (18082306)	20.31255 (18082306)	20.66113 (18081306)	20.60355 (18082306)	22.32004 (18082306)
4498885.0	14.69429 (18082306)	18.35081 (18082306)	18.62057 (18081306)	18.73905 (18082306)	19.47105 (18082306)

*** AERMOD - VERSION 18081 *** *** THESS *** 05/31/21
 *** AERMET - VERSION 19191 *** *** 09/13/21

CO



PM10



*** THE 1ST HIGHEST 3-HR AVERAGE CONCENTRATION VALUES FOR SOURCE GROUP: ALL ***
 INCLUDING SOURCE(S): PARSA1 , PARSA2 , PARSA3 , PARSA4 ,

*** NETWORK ID: UCART3 , NETWORK TYPE: GRIDCART ***

** CONE OF PM10 IN MICROGRAMS/M**3 **

Y-COORD (METERS)	661192.00	661192.00	661192.00	661192.00	661192.00
4501005.0	6.26522 (18102405)	5.32184m (18011121)	2.96775 (18011924)	3.40168 (18092021)	2.21151 (18011024)
4501715.0	7.76107m (18051121)	4.87190 (18101005)	4.81707 (18011524)	4.00956 (18092021)	2.57638 (18011524)
4501585.0	5.97830m (18072903)	7.14159 (18102405)	7.45814 (18011524)	7.06899m (18060103)	4.84804 (18011524)
4501455.0	7.47600 (18011524)	7.82407m (18011121)	8.86585 (18011524)	8.90960m (18060103)	7.55956 (18011524)
4501325.0	11.07759 (18101106)	8.75563 (18011524)	7.29410m (18011121)	9.01913m (18060103)	8.07085 (18011524)
4501195.0	15.51616 (18101106)	12.42788 (18101106)	9.15455 (18011524)	12.40647 (18091021)	8.29700m (18011121)
4500965.0	14.83964 (18101106)	17.81432 (18101106)	15.55778 (18011524)	14.08897 (18011524)	11.22687m (18011021)
4500835.0	20.76240 (18102406)	17.58397 (18042105)	22.79427 (18101106)	14.15006 (18100206)	12.54266 (18101106)
4500605.0	31.64536 (18102406)	27.80978 (18102406)	38.80362 (18102406)	23.75645 (18101106)	18.46205 (18100206)
4500475.0	47.71800 (18001505)	55.55026 (18001505)	65.67277 (18102406)	55.20149 (18101406)	34.57656 (18102406)
4500345.0	58.77602 (18040406)	59.27015 (18040406)	98.72722 (18001505)	41.79065 (18001505)	48.00918 (18102406)
4500215.0	28.03367 (18070524)	36.49211 (18101824)	66.48257 (18101824)	41.77553 (18040406)	40.50533 (18040406)
4500085.0	25.56850m (18072206)	25.75669m (18072206)	31.48270m (18072206)	20.26999m (18072206)	24.43392m (18072206)
4499955.0	1.17030m (18111424)	1.59382m (18052224)	1.78201 (18051124)	5.00192 (18002306)	2.62865 (18002306)
4499785.0	1.03707m (18052224)	1.09228m (18052224)	1.16851m (18052224)	1.98647 (18002306)	1.60197 (18002306)
4499655.0	0.84975m (18052224)	0.84059m (18052224)	0.80022 (18051124)	1.49151 (18002306)	1.19173 (18002306)
4499485.0	0.67033 (18051124)	0.73069 (18051124)	0.78758 (18051124)	1.39566 (18002306)	0.98454 (18002306)
4499355.0	0.62657 (18051124)	0.65384 (18051124)	0.60588 (18051124)	0.99322 (18002306)	0.85951 (18002306)
4499185.0	0.55846 (18051124)	0.53002 (18051124)	0.40492 (18051124)	0.84588 (18002306)	0.77319 (18002306)
4499055.0	0.43694 (18051124)	0.34334 (18051124)	0.33670m (18011824)	0.73400 (18002306)	0.70757 (18002306)
4498885.0	0.29598 (18051124)	0.20660m (18011824)	0.31333m (18011824)	0.64661 (18002306)	0.65434 (18002306)

*** AERMOD - VERSION 18081 *** ** THESIS *** 05/31/21
 *** AERMOD - VERSION 19191 *** ** 09:35:29
 PAGE 18

AIR QUALITY STANDARDS

AMBIENT AIR QUALITY STANDARDS FOR THE EUROPEAN UNION

Table 54. AMBIENT AIR QUALITY STANDARDS FOR THE EUROPEAN UNION

Pollutant	Concentration	Averaging period	Legal nature	Permitted exceedences each year
Fine particles (PM _{2.5})	25 µg/m ³ ***	1 year	Target value entered into force 1.1.2010 Limit value enters into force 1.1.2025	n/a
Sulphur dioxide (SO ₂)	350 µg/m ³	1 hour	Limit value entered into force 1.1.2005	24
	125 µg/m ³	24 hours	Limit value entered into force 1.1.2005	3
Nitrogen dioxide (NO ₂)	200 µg/m ³	1 hour	Limit value entered into force 1.1.2020	18
	40 µg/m ³	1 year	Limit value entered into force 1.1.2010*	n/a
PM ₁₀	50 µg/m ³	24 hours	Limit value entered into force 1.1.2005**	35
	40 µg/m ³	1 year	Limit value entered into force 1.1.2005**	n/a
Lead (Pb)	0.5 µg/m ³	1 year	Limit value entered into force 1.1.2005 (or 1.1.2010 in the immediate vicinity of specific, notified industrial sources; and a 1.0 µg/m ³ limit value applied from 1.1.2005 to 31.12.2009)	n/a
Carbon monoxide (CO)	10 mg/m ³	Maximum daily 8 hour mean	Limit value entered into force 1.1.2005	n/a
Benzene	5 µg/m ³	1 year	Limit value entered into force 1.1.2020***	n/a
Ozone	120 µg/m ³	Maximum daily 8 hour mean	Target value entered into force 1.1.2010	25 days averaged over 3 years
Arsenic (As)	6 ng/m ³	1 year	Target value enters into force 31.12.2012	n/a
Cadmium (Cd)	5 ng/m ³	1 year	Target value enters into force 31.12.2012	n/a
Nickel (Ni)	20 ng/m ³	1 year	Target value enters into force 31.12.2012	n/a
Polycyclic Aromatic Hydrocarbons	1 ng/m ³ (expressed as concentration of Benzo(a)pyrene)	1 year	Target value enters into force 31.12.2012	n/a

Appendix F –PEI Report example (PDF)

Port Environmental Index Report

Calculation Period

- **Initial Date:** 2020-11-30T22:00:00.000Z
- **Final Date:** 2020-12-31T22:00:00.000Z

Configurations

- **Normalization method:** DISTANCE TO A REFERENCE PORT
- **Weighting Method:** EQUAL WEIGHTING
- **Aggregation Method:** ARITHMETIC
- **Update Strategy:** REPLICATE LAST VALUE
- **Data imputation approach:** HOT DECK

Global values

PEI value: 0.44202188

RR value: 66.49 %

PEI Indices values

SEI: 0.688853 **TEI:** 0.4390271 **GEI:** 0.19818549

RR Indices values

SRR: 83.46 % **TRR:** 85.42 % **GRR:** 30.61 %

eKPIs values

Origin	Subindex	eKPI	Value	Unit	Normalised value
SEI	Air Emission	CO2	0.7478479	kg/ton	0.06653889
SEI	Air Emission	NOX	0.0152900675	kg/ton	0.06653993
SEI	Air Emission	PM10	0.001537538	kg/ton	0.06654795

SEI	Air Emission	PM2.5	0.001441018	kg/ton	0.06654671
SEI	Air Emission	SO2	0.012420504	kg/ton	0.06653916
SEI	Air Emission	HC	4.96819E-4	kg/ton	0.0665836
SEI	Air Emission	CO	0.00117969	kg/ton	0.06654714
SEI	Air Emission	N2O	3.2173E-5	kg/ton	0.06651539
SEI	Air Emission	CH4	1.0724E-5	kg/ton	0.06667288
SEI	Waste	Passively fished waste	0.0	kg/ton	1.0
SEI	Waste	E-waste	0.0	kg/ton	1.0
SEI	Waste	Cargo residues -harmful-	0.0	kg/ton	1.0
SEI	Wastewater	Oily bilge water	0.0	kg/ton	1.0
SEI	Wastewater	Oily residues -sludge-	0.0	kg/ton	1.0
SEI	Wastewater	Oily tank washings	0.0	kg/ton	1.0
SEI	Wastewater	Dirty ballast water	0.0	kg/ton	1.0
SEI	Wastewater	Scale and sludge from tank cleaning	0.0	kg/ton	1.0
SEI	Wastewater	Other - oil	0.0	kg/ton	1.0
SEI	Wastewater	NLS - type X	0.0	kg/ton	1.0
SEI	Wastewater	NLS - type Y	0.0	kg/ton	1.0
SEI	Wastewater	NLS - type Z	0.0	kg/ton	1.0
SEI	Wastewater	NLS - other	0.0	kg/ton	1.0
SEI	Wastewater	Sewage	0.0	kg/ton	1.0
TEI	Air Emission	CO2	356.17334	g/ton	0.7093981
TEI	Air Emission	NOX	4.387214	g/ton	0.799797
TEI	Air Emission	PM10	0.17646797	g/ton	0.8098388
TEI	Air Emission	PM2.5	0.12518586	g/ton	0.80847776
TEI	Air Emission	SO2	0.57104355	g/ton	0.8128368
TEI	Air Emission	HC	0.37162748	g/ton	0.8085522
TEI	Air Emission	CO	0.6955883	g/ton	0.77622485
TEI	Air Emission	N2O	0.00887605	g/ton	0.8057394
TEI	Air Emission	CH4	0.044258993	g/ton	0.7387356
TEI	Waste	Municipal solid waste	739705.44	kg/ton	2.0329056E-8
TEI	Waste	Inert waste	0.0284807	kg/ton	0.27365163
TEI	Waste	Hazardous waste	0.003519851	kg/ton	0.003977441
GEI	Noise	Iden	79.31922	dB	0.02523179
GEI	Noise	hight	71.26399	dB	0.0052320496
GEI	Light	Light pollution	163332.61	ucd/m2	0.38113907

Reliability Rating values

Data origin	Subindex	Piece of data	Optimal retrieval way	Current retrieval way	Reliability Rating	Aggregated RR (subindex)	Aggregated RR(origin)
	Air	IMO number and	Real Time				

SRR	Emission	gross tonnage of ships	API	Periodic API	80.55 %	67.96 %	83.46 %
SRR	Air Emission	Main and auxiliary engine of ships	Periodic API	Average Value From Literature	36.84 %	67.96 %	83.46 %
SRR	Air Emission	Berth and maneuvering time of ships	Sensors	Periodic API	86.48 %	67.96 %	83.46 %
SRR	Waste	MARPOL annexes about waste	Pixel Proxy Tool	Periodic API	97.14 %	97.14 %	83.46 %
SRR	Wastewater	MARPOL annexes about wastewater	Pixel Proxy Tool	Periodic API	85.29 %	85.29 %	83.46 %
TRR	Air Emission	Emissions produced by terminal machinery	Sensors	Periodic API	73.47 %	73.47 %	85.42 %
TRR	Waste	Waste produced by the terminals	Pixel Proxy Tool	Periodic API	97.36 %	97.36 %	85.42 %
GRR	Noise	Noise values Lden - Lnight - Leq	Sensors	Average Value From Literature	34.68 %	34.68 %	30.61 %
GRR	Light pollution	Luminosity-lux	Sensors	Average Value From Literature	26.53 %	26.53 %	30.61 %

Recommendations

- Installation of sea water scrubbers to control exhaust gases from the ships' auxiliary engines.
- Use of Green fees based on ship fuel consumption and specifications and validate the ship engine performance compliance with international regulations regarding combustion emissions. Promote the use of low sulphur fuel in port.
- Provide on-shore power supply ('cold ironing').
- Implementation of systems for offering pre-booked port slots and for the optimisation of cargo handling equipment allocation, together with the installation of an automated mooring system.
- Implement voluntary programmes for vessel speed reduction or/and the virtual arrival concept to manage ports' operational delays.
- Use of light emitting diode (LED) lights in buildings, docks, yards, storages, warehouses, and tugs, an automatic lighting controls and sensors and high mast lighting to reduce light loss, Buildings insulation using green roofs, and new buildings construction with designs that minimise cooling demand and heat loss, Energy efficient Control of HVAC in buildings and warehouses, Painting external walls white, cleaning lamps, cold storage insulation and curtains are suggested for warehouses and storages, Reefer monitoring systems and reefer sun protection roofs, preferably combined with photovoltaic panels. Gaps between adjacent reefers can be isolated from surrounding air by elastic seals, Wall and roofing insulation on storage tanks and pipelines in liquid bulk terminals, Design port layouts and facilities to minimize travel distances and transfer points and to avoid /minimize re-storage and re-shuffling of cargo.
- Implementation of programmes for improving employee's awareness of energy efficiency through environmental training and eco driving lessons.

Figure 53. PEI Report example